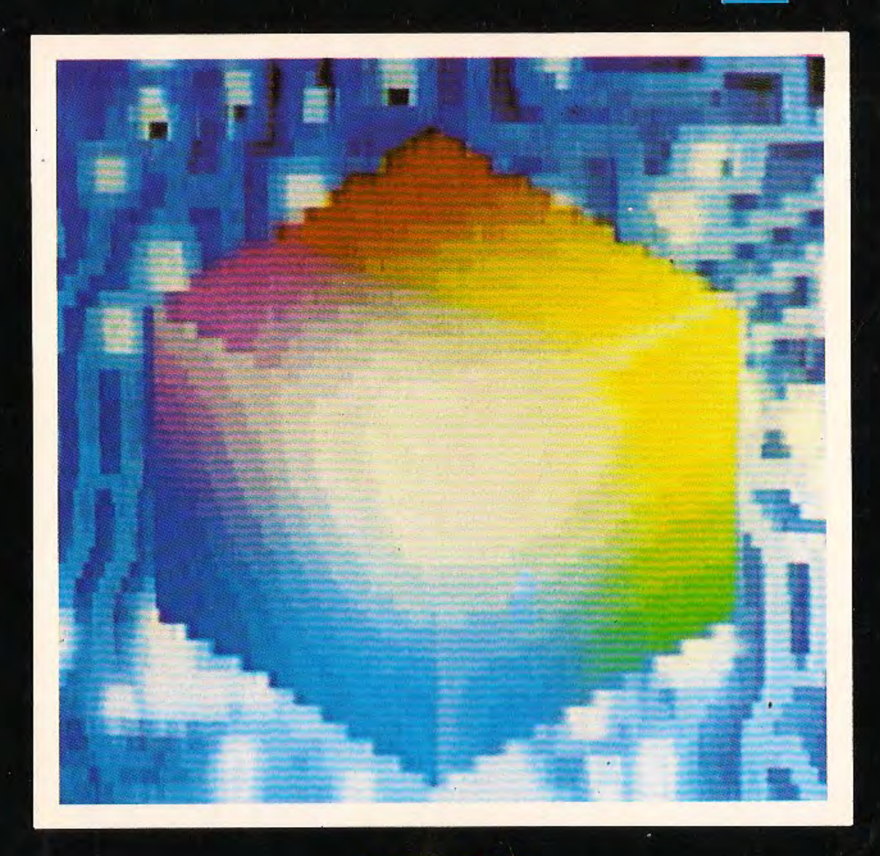
BIBLIOTECA BASICA INFORMATICA

SISTEMAS OPERATIVOS Y SOFTWARE DE BASE 10



INGELEK

BIBLIOTECA BASICA INFORMATICA

SISTEMAS OPERATIVOS Y SOFTWARE DE BASE 10

INGELEK

Director editor:
Antonio M. Ferrer Abelló.

Director de producción:
Vicente Robles.

Coordinador y supervisión técnica:
Enrique Monsalve.

Colaboradores:
Angel Segado.
Patricia Mordini.
Margarita Caffaratto.
Marina Caffaratto.
Francisco Ruiz.
Jorge Juan Monsalve.
Beatriz Tercero.
Fernando Ruiz.
Casimiro Zaragoza.

Diseño:
Bravo/Lofish.

Dibujos:
José Ochoa.

ISBN del tomo: 84-85831-43-8
ISBN de la obra: 84-85831-31-4
Fotocomposición: Pérez Díaz, S. A.
Imprime: Héroes, S. A.
Depósito Legal: M-349-1986

INDICE

PROLOGO

Antes de decidirnos a escribir acerca de los sistemas operativos para ordenadores domésticos y personales de manera que fuese comprensible para personas no expertas, nos lo pensamos bastante, ya que el tema es, sin duda alguna, uno de los más complejos en el campo de la Informática, tratado únicamente en facultades de Informática e Ingeniería en cursos altamente especializados. Por otra parte, la gran difusión de los ordenadores ha obligado a las industrias a crear unos sistemas operativos que se adapten a las características de sus máquinas y a las exigencias de unos usuarios que no son profesionales de grandes centros de cálculo, sino estudiantes y aficionados, comerciantes y personas de todo tipo.

Así, las características de los sistemas operativos han cambiado en relación con aquellas otras de los grandes equipos. Se ha ido difundiendo la idea de que estos aparatos tan sofisticados necesitan ser día a día más versátiles, es decir, fáciles de operar por parte de un número cada vez mayor de personas.

De todos modos, iremos viendo a lo largo de este libro cómo los sistemas operativos de los ordenadores personales repiten en el tiempo el desarrollo experimentado por sus hermanos mayores. En efecto, dado que los progresos tecnológicos referentes a los circuitos integrados están reduciendo cada vez más la diferencia entre "micro" y "mini" ordenador, así los sistemas operativos asumen en el campo del ordenador personal los logros obtenidos en los grandes sistemas, trasvasando a los "pequeños" muchas de tales experiencias. En este aspecto aparece el Unix, sistema operativo que en apariencia empieza a predominar, espe-

cialmente en el rango de los microordenadores profesionales. Este y otros hechos inducen a considerar no utópico el suponer que, en un futuro no muy lejano, las diferencias entre ambas categorías se reducirán hasta desaparecer.

Este libro, lo repetimos de nuevo, no está destinado a los estudiantes universitarios de Informática, sino a aquellas personas que deseen acercarse a este tema sin verse obligadas a leer toneladas de libros en inglés, y sin renunciar al mismo tiempo a un tratamiento riguroso.

A propósito del inglés, éste ha sido otro de los problemas que se nos han planteado, ya que el lenguaje utilizado en Informática está lleno de vocablos americanos, a menudo imposibles de traducir por tratarse de neologismos, incluso en su propio idioma de origen. Asistimos así a la difusión de una extraña lengua con palabras del tipo "digitalizar" (de digit). Por otra parte, está claro que el idioma normalmente utilizado en Informática es el inglés, y no sabemos hasta qué punto sería interesante traducir cada palabra. El único país del mundo en donde se ha intentado hacer una operación de este tipo (indudablemente de mucho mérito), ha sido Francia, pero con resultados no tan satisfactorios como los esperados. En efecto, ha resultado ser un lenguaje poco práctico, ya que aquellas personas interesadas en el campo de los ordenadores se ven obligadas a estudiar dos idiomas: el americano técnico, para poder seguir los desarrollos y avances en el tema que les interesa, siempre en publicaciones en inglés, y el "francés-informatizado" para entenderse con los compatriotas.

Nosotros hemos intentado seguir una vía intermedia, utilizando aquellas palabras en español que ya están integradas dentro del vocabulario corriente, y los vocablos ingleses cuando resulte imposible o fuera de lugar la traducción (por ejemplo, con bit, byte, etc.). Hemos traducido algunos términos cuando son indistintamente utilizados en inglés y en español.

Vamos a tratar ahora la cuestión de la organización del libro, cuya materia, como ya dijimos, es amplia y compleja. Nuestra intención no es la de presentar los problemas de Dijsktra y Brinch Hansen (importantes "cerebros" en la materia) sobre procesos concurrentes y multiprogramación, sino facilitar el trabajo de aquellas personas que se acercan por primera vez a estos temas en el mundillo de los ordenadores personales.

En el primer capítulo veremos cuál ha sido la historia de los ordenadores y de los sistemas operativos desde su nacimiento hasta hoy, para, una vez definidas las características que éstos han asumido en los ordenadores personales, examinar los aspectos fundamentales de los sistemas operativos más difundidos.

El segundo capítulo presenta aquello que hoy en día es uno de los aspectos de mayor interés: la cuestión de la memoria, haciendo también referencia a la organización adoptada por el IBM-PC.

En la tercera parte están tratados los dispositivos para el almacenamiento de grandes cantidades de datos: cintas y discos. Examinamos de los mismos sus características en cuanto al diseño se refiere, y los aspectos de gestión que afectan e interesan al sistema operativo.

En el último capítulo nos referimos a aquellos programas que, si bien no forman parte del sistema operativo, sí constituyen, digámoslo así, su natural expansión formando con él el llamado "software de base", parte esencial en el funcionamiento y uso eficaz del ordenador. Estamos hablando de los Editores, Ensambladores, Intérpretes, Compiladores para los lenguajes de alto nivel, y hasta de los Word Processor y otros que cruzan la frontera, entrando en el área propiamente de aplicaciones...

A aquellas personas que, una vez concluida la lectura, quieran saber algo más de esta materia, les aconsejamos consulten la bibliografía que incluimos o que se dirijan a alguna librería técnica.

CAPITULO I

SISTEMAS OPERATIVOS: NACIMIENTO, HISTORIA Y MODELOS

El sistema operativo, es el instrumento indispensable para hacer del ordenador un objeto vivo y, sobre todo, útil para nosotros. En efecto, bajo este nombre se agrupan todos aquellos programas que permiten a los usuarios la utilización de ese embrollo de cables y circuitos, que de otra forma serían difíciles de controlar.

Formalmente, un sistema operativo se define como un conjunto de procedimientos, manuales y automáticos, que permiten a un grupo de usuarios compartir una instalación de ordenador eficazmente.

Si se trata de una instalación particular que no va a ser compartida, utilizada sólo por un único usuario se amplia la definición anterior diciendo que los sistemas operativos posibilitan al usuario la realización de determinadas operaciones (por ejemplo, las de entrada/salida) de forma cómoda y eficaz.

Pocas personas saben que, en el origen de la historia de los ordenadores, es decir, hace unos cuarenta años, los sistemas operativos no existían; así, la introducción de un programa para ser ejecutado se convertía en un increíble esfuerzo que sólo podía ser llevado a cabo por muy pocos expertos. El tiempo requerido para meter un programa en aquellos mastodontes de escaso cerebro superaba con mucho el de ejecución, de modo que resultaba poco provechosa la utilización de ordenadores para la resolución de problemas prácticos.

Como es fácil suponer, el nacimiento de los sistemas operativos y su posterior desarrollo han sido decisivos para lograr el nivel de difusión, cada vez mayor, de los ordenadores. La tendencia más generalizada en nuestros días es la de sistemas operati-

vos "amigables" ("friendly"), es decir, orientados hacia una comprensión y manejo inmediatos por parte del usuario, sin necesidad de grandes esfuerzos mentales.

Creemos que la mejor forma de acercamiento a este tema es el conocer primero algo de su historia en relación con el desarrollo de los ordenadores, desde los años cuarenta hasta nuestros días. En efecto, hay una clara tendencia por parte de los ordenadores personales de seguir los pasos de sus hermanos mayores hasta en los más mínimos detalles. Así los sistemas operativos de los ordenadores personales han recorrido hasta el día de hoy aproximadamente el mismo camino que los grandes ordenadores entre los años cuarenta y sesenta. La evolución de estos en años sucesivos nos dirá, por tanto, con mucha probabilidad, cuál será el desarrollo que se verificará en el campo de la microinformática en los próximos años.

Un poco de historia

La evolución de los sistemas operativos sigue muy de cerca a la de los ordenadores. Estos han ido desarrollándose según el avance de la tecnología, desde los primeros procesadores a válvulas hasta los actuales, basados en circuitos a grandísima escala de integración (VLSI, Very Large Scale of Integration). La historia de los sistemas operativos está catalogada por generaciones (aproximadamente las mismas de los ordenadores): desde la generación cero, en los años cuarenta, cuando existían los procesadores, pero aún no sistemas operativos dignos de este nombre, a la cuarta generación, muy sofisticada, utilizada hoy en día en los sistemas más avanzados.

Antes de la década de los cincuenta los sistemas operativos estaban anclados en la prehistoria: los procesadores estaban constituidos por enormes cajas de válvulas que presentaban en sus paneles una serie de interruptores. Por medio de éstos el operador introducía en el ordenador una serie de "ceros" y "unos"; iniciaba el proceso y examinaba, por medio de bombillas, el contenido de la memoria para obtener los resultados. Como es fácil de suponer, se trataba de un sistema largo, complejo y que hacía muy dificultoso el manejo de la máquina.

En el comienzo de los años cincuenta surgieron los primeros, rudimentarios pero ya útiles, sistemas operativos. Su objetivo era reducir los tiempos muertos necesarios entre la ejecución de un programa y otro, ya que hasta aquel momento la mayor parte del tiempo se perdía en las fases intermedias, durante las cuales el programa era introducido en el sistema, y al final del proceso, cuando había que extraer los resultados. El programa se "cargaba" entonces por medio de un lector de tarjetas, ocupando ya en esta fase la memoria completa. Al final del proceso el operador debía cargar otro programa para proceder al examen de la memoria, remover cintas, tarjetas y datos impresos. Sólo en ese momento el ordenador podía aceptar otro programa, repitiendo desde el comienzo las diferentes fases (Fig. 1).

Para reducir estos tiempos muertos fueron creados los sistemas operativos por lotes (o en "batch"), en los que los programas eran tratados por grupos (lotes) en vez de individualmente. Así surgen los llamados BATCH, pequeños ordenadores auxiliares que se encargaban de leer las tarjetas de los distintos programas y de transferir su contenido a una cinta magnética. La función del sistema operativo consistía en cargar en memoria un programa de la cinta y ejecutarlo: al finalizar se realizaba el salto a una dirección de memoria desde donde reasumía el control el sistema operativo, que cargaba el siguiente programa y lo ejecutaba. De esta manera el tiempo entre un trabajo y el otro era netamente inferior (Fig. 2).

Pronto se impuso otro concepto fundamental, que sigue siendo válido en nuestros días: la definición de los dispositivos de entrada y salida por medio de nombres lógicos. Así no era ya necesario conocer con todo detalle estos sub-procesos, sino que se podía genéricamente hacer referencia a un "lector de tarjetas" para la entrada o a una "impresora" para la salida. Ahora era el sistema operativo el que se preocupaba de hacer la función de "Interface" entre los programas y el exterior, gacias a los IOCS (Input/Output Control System), grupos de programas para gestionar los periféricos en todos sus detalles. Hoy en día este tipo de programas ha sido desarrollado y perfeccionado, permitiendo al usuario toda una serie de operaciones sofisticadas, independientemente de cómo las realicen los periféricos. Surgen los DASD dispositivos de almacenamiento de acceso directo (Direct Access Storage Devices) (Fig. 3).

En la década de los sesenta, con la segunda generación, asistimos al nacimiento de los sistemas de tiempo compartido: se trata del inicio de la multiprogramación y del sistema multiprocesa-



Figura 1.—Proceso que permitían los primeros sistemas operativos.

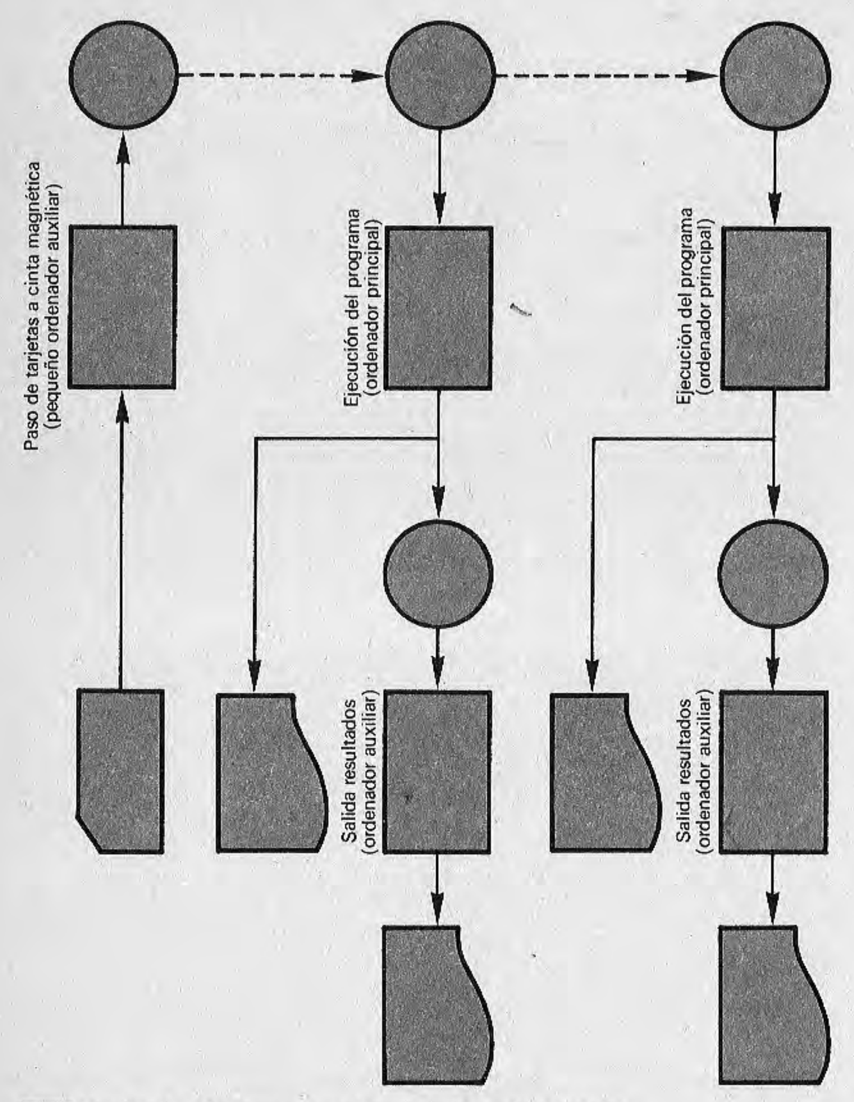


Figura 2.—Sistema operativo por lotes (en batch).

dor. Veamos qué significan estos conceptos que, si bien llevan el mismo resultado externo, se basan sobre pilares completamente diferentes.

La sincronización entre el procesador central y los dispositivos periféricos se resolvió mediante las interrupciones. Una in-

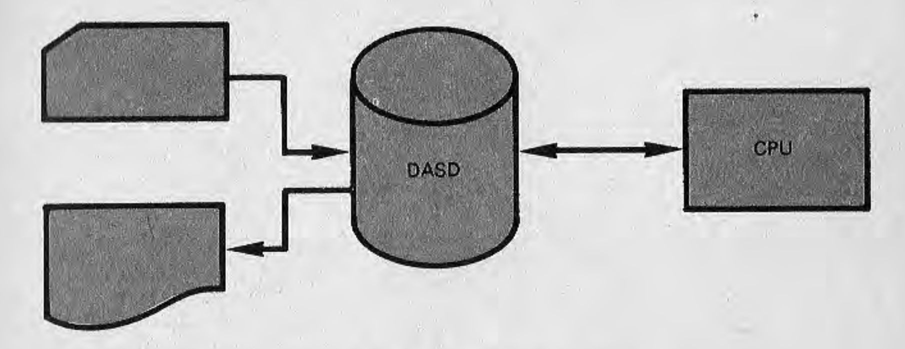


Figura 3.—Situación de los DASD en el sistema.

terrupción es una señal de sincronización enviada por un dispositivo periférico a un registro accesible para el procesador central. Este registro es examinado por el procesador central después de la ejecución de cada instrucción. Cuando su contenido señala una interrupción, el procesador central suspende la ejecución del programa en curso y da comienzo la de otro ("atiende la interrupción"). Esta técnica hace posible el funcionamiento concurrente de un procesador central y sus dispositivos periféricos. La técnica de programación usada para manejar el funcionamiento concurrente del ordenador se denomina MULTIPROGRAMACIÓN.

Con la multiprogramación se hallan presentes en la memoria del ordenador varios programas a la vez. El procesador, único, conmuta su funcionamiento —su atención— continuamente de un programa a otro. Al ser los tiempos de conmutación muy pequeños en relación con los de reacción de los seres humanos, los usuarios no advierten una diferencia sensible en el funcionamien-

El procedimiento de tiempo compartido, de todos modos, hace más lentas las prestaciones, aunque es muy difícil que el usuario se dé cuenta de ello.

En los sistemas multiprocesadores, en cambio, se hallan varios procesadores simúltáneamente en el mismo ordenador: de esta forma las prestaciones quedan incrementadas sobremanera.

En el mismo período ve la luz otro concepto fundamental de la filosofía multiusuario: la partición de tiempo, más conocida como "timesharing". Por fin los usuarios podían establecer un diálogo con el ordenador utilizando un terminal dotado de un teclado, por medio del cual era posible enviar mensajes, y otro periférico por donde recibir las respuestas. Se hacía así posible un funcionamiento interactivo en el que el usuario hacía una petición, el ordenador la aceptaba, la ejecutaba y enviaba la respuesta al terminal del usuario (Fig. 4).

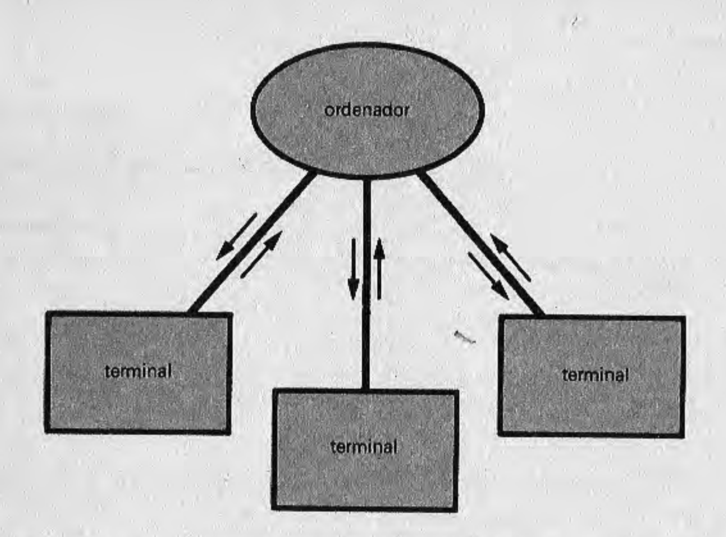


Figura 4.—Con los sistemas operativos de timesharing el ordenador observa todos los terminales. Si alguno envía una demanda, la procesa y devuelve los resultados al mismo terminal, retornando a la situación de "escucha".

Este funcionamiento ha permitido desarrollos formidables en el campo de la Informática. Basta pensar en el proceso de elaboración y verificación de un programa. Antes del advenimiento de estos sistemas operativos el usuario debía perforar un buen número de tarjetas (una para cada instrucción), introducirlas luego en el lector de tarjetas y, cuando llegaba el momento (tras algunas horas), retiraba un listado en donde estaban señalados los errores. Así, antes de tener una versión definitiva del programa y lograr que funcionara transcurría muchísimo tiempo. Con los sistemas operativos Timesharing, en cambio, cada usuario podía escribir, compilar y ejecutar los programas de una manera mucho más cómoda y rápida.

Pero, como siempre, también está la otra cara de la moneda: los nuevos sistemas operativos, para poder ofrecer estos servicios tan útiles, además de haber costado grandes esfuerzos su implementación, requerían muchísima memoria para poder funcionar. En efecto, como es fácil comprender, la multiprogramación requiere una gran cantidad de funciones, todo un software que consume mucha memoria.

Esta técnica nació al comprobarse que la unidad central del ordenador permanecía inactiva durante la fase de entrada/salida de datos, al ser esta operación muy lenta en comparación con la velocidad del procesador central. Se pensó entonces que durante estas actividades (que se podían realizar con dispositivos autocontrolados situados en los periféricos) la atención del ordenador podía ser desviada hacia otro programa para ahorrar tiempo, reanudando la ejecución del programa original al término de su fase de entrada/salida (Fig. 5). Naturalmente, puede suceder que el 2.º programa inicie también una actividad de entrada/salida, de modo que el sistema operativo debe encargarse entonces de ceder el control a un tercer programa. Esto puede dar lugar a una verdadera cadena de actividades, obteniendo así el llamado overlapping (superposición) de actividades de cálculo y de Input/Output, base del ahorro del tiempo.

La operación de conmutación de la unidad central de un programa a otro es muy delicada, ya que debe lograr salvar toda la

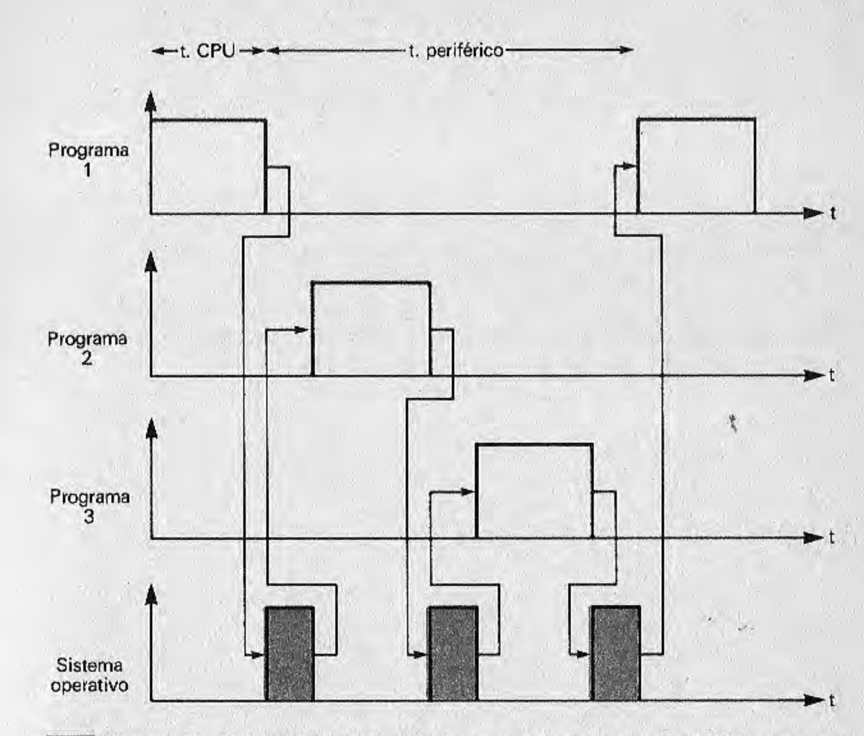


Figura 5.—Sucesión temporal de tres programas en multiprogramación. Durante el tiempo de actividad del periférico pueden trabajar sucesivamente (en "overlapping") otros programas: la línea discontinua indica el tiempo de acción del sistema operativo, necesario para salvar los parámetros de un programa, antes de pasar a otro.

información en posesión del procesador, incluido el contenido de los registros y las direcciones de memoria en que está trabajando, de forma que se pueda recuperar después en el momento de restaurar las condiciones de funcionamiento.

Está claro que cuando el número de programas "simultáneamente" en proceso (es decir, en actividad conmutada) se hace considerable, la cantidad de trabajo que el sistema operativo debe efectuar para salvar y restablecer los contextos de las distintas actividades adquiere grandes proporciones. Se trata de un límite para este tipo de organización: en efecto, es necesario no superar nunca una cierta carga para el sistema, pues de otra forma éste pierde completamente su eficacia, llegando a una situación en la cual todo el tiempo-máquina se consume en ejecutar operaciones de cambio de contexto.

IBM 360 y Unix

A caballo entre los años sesenta y setenta se realizan los mayores esfuerzos en la mejora de los sistemas operativos. Los productos más importantes son el Sistema Operativo de la familia IBM 360 y el Unix. Totalmente diferentes en cuanto a planteamientos y objetivos se refiere, el primero ha dominado la escena de los grandes ordenadores hasta hace poco tiempo, y el segundo está consiguiéndolo ahora.

La tercera generación de ordenadores tenía como fin el de estar destinada a un campo de aplicaciones lo más extendido posible. La firma IBM pensaba realizar, con la familia 360, un sistema capaz de satisfacer las exigencias más dispares del usuario. Se quería ofrecer un ordenador que el usuario pudiera utilizar durante un gran número de años, aun con exigencias diferentes de las originales, sin verse obligado a cambiar por ello la estructura completa.

Esta forma de pensar ha sido muy importante en la historia de los ordenadores y ha condicionado su evolución hasta nuestros días. Además, para convencer a los compradores en posesión de otros ordenadores a pasarse a los IBM, el sistema operativo del 360 era capaz, por medio de simuladores y emuladores, de ejecutar directamente programas realizados en otros ordenadores.

El nacimiento de este sistema fue como una bomba en el mercado internacional. Las firmas competidoras debieron tomar pronto medidas, cada una de un modo diferente. La RCA, como otras, pretendió copiar el sistema operativo del 360 para venderlo a un precio inferior. Este intento supuso un fracaso tan clamoroso que

provocó la retirada de la RCA del mundo de la Informática con un déficit superior a los 500 millones de dólares.

Otras compañías, como la General Electric y la Burroughs, trataron de realizar un sistema operativo no compatible con el 360, pero más eficiente y potente. Este proyecto también resultó ser demasiado ambicioso, por lo que la General Electric tuvo, a su vez, que abandonar el mundo de los ordenadores.

Firmas como Control Data, Univac y Honeywell, en cambio, se limitaron a desarrollar sistemas más sencillos y similares a los más antiguos de IBM, llegando a convencer así a aquellos usuarios que no se decidían a pasar a la categoría de los 360.

Por otra parte, las esperanzas de IBM de realizar un único sistema operativo para toda la serie 360 quedaron también frustradas debido a la gran cantidad de necesidades diferentes de los usuarios. Fueron así realizados cuatro sistemas operativos, acordes a las distintas dimensiones de los ordenadores:

- — DOS/360 para los más pequeños,
- — OS/MFT y OS/MVT para medianos/grandes,
- — CP-67/CMS para los más potentes.

Poseían como característica común la de ofrecer a los usuarios un número notable de programas de soporte que permitían una utilización más simple y provechosa del ordenador.

Pocos años después del nacimiento de la serie 360 salía a la luz un sistema operativo basado en una filosofía totalmente diferente: el Unix. Concebido por un grupo de investigadores del Bell Laboratory que habían participado en el estudio de un sistema operativo destinado a competir con el 360, el Multics, el Unix se proyectó con el fin de ofrecer un sistema operativo económico en el cual la gestión de programas en disco fuese independiente del dispositivo. El Unix oponía así su sencillez y eficacia a la complejidad y enormidad del 360 IBM.

Desarrollado en un ambiente universitario, posee características mucho más avanzadas respecto a las de la mayor parte de los sistemas operativos. Su ventaja más interesante bajo un punto de vista industrial es la de haber sido diseñado de forma que pueda ser adaptable a las más diversas arquitecturas, a las que sólo exige que estén dotadas al menos de 256 Kbytes de memoria y de un microprocesador de 16 o más bits. Este hecho permite suponer que el Unix será uno de los sistemas operativos más difundidos a nivel de minis y medianos ordenadores, además de ser el candidato con mayores posibilidades para la próxima generación de ordenadores personales.

La actual generación de sistemas operativos está claramente influida por el fenómeno más importante de estos últimos años: la

difusión de los microprocesadores. Gracias a estos dispositivos, aquellos ordenadores enormes que hasta hace pocos años costaban billones, hoy en día pueden ser fabricados por pocas decenas de millones. Incluso la CPU del IBM 360 ha sido reducida a un circuito integrado de unas dimensiones de pocos milímetros cuadrados.

De este modo, los sistemas operativos han debido adaptarse al hecho de que, con precios decididamente razonables, se pueden obtener hoy prestaciones impensables en otro tiempo. En efecto, ya hay disponibles redes de ordenadores en donde el sistema operativo debe encargarse de la gestión atenta e inteligente de una enorme cantidad de dispositivos (ordenadores, periféricos y unidades de memoria).

Una de las novedades más interesantes de los sistemas operativos modernos es la técnica de gestión de Memoria Virtual, gracias a la cual el usuario puede utilizar programas de dimensiones bastante superiores a la cantidad de memoria interna disponible. Este "milagro" se debe a que, a su debido tiempo, son cargadas en el ordenador las "páginas" de software y de datos necesarios en ese instante, tomadas de las memorias de masa externas. De todo esto se encarga el sistema operativo, sin que el usuario deba poseer ningún tipo de noción acerca de cómo está constituido internamente el ordenador. Es decir, por fin se está produciendo aquel que era el propósito de los estudiosos de sistemas operativos desde sus orígenes: lograr que el usuario no tenga por qué ser un experto de electrónica o informática, sino tan sólo lo que su nombre indica, un usuario. De esta manera, los modernos ordenadores se presentan con una serie de "menús" (pantallas con posibles opciones) que permiten elegir las funciones que desea ejecutar, y será el servicial sistema operativo quien se preocupará de manejar todos los dispositivos de hard y soft que la operación requiera.

Los sistemas operativos en los ordenadores personales

Gracias a los avances logrados con el desarrollo de la tecnología y a la ley del mercado, los sistemas operativos de los ordenadores personales están poniéndose rápidamente al nivel de sus hermanos mayores. Desde las primeras placas como de juguete, en donde una serie de interruptores encendían unos LEDs rojos y verdes, hemos pasado a placas de microprocesadores bastante más elaboradas.

Hasta hace poco ningún ordenador personal estaba dotado de sistema operativo, sino sólo de un "Monitor". Y no se trata de la pantalla de visualización, sino de un programa, escrito a propósito para cada placa de microprocesador, que se halla almacenada en una EPROM (Erosable Programable Read Only Memory, memoria de sólo lectura programable y borrable) y que cumple sólo algunas operaciones, de las más elementales, de un sistema operativo, permitiendo la utilización de un terminal de vídeo, un teclado, una impresora, un casete de cintas magnéticas y, a veces, una unidad de disco. En efecto, transportar un sistema operativo desde un gran ordenador resultaba demasiado costoso para los primeros microprocesadores, ya que los sistemas operativos requieren una gran cantidad de memoria y, al menos, una unidad de disco, elementos éstos que comportan un aumento considerable en el precio del equipo. El problema fue inicialmente resuelto por medio de las memorias EPROM, ya que estos circuitos integrados permiten implementar y conservar permanentemente en memoria un cierto número de programas de utilidad a precios razonables.

Las funciones de estos "monitores" se limitaban sólo a las esenciales para el funcionamiento del ordenador, es decir: escritura y lectura de memoria y registros en forma hexadecimal, posibilidad de hacer ejecutar y parar los programas, manejo de pantalla, teclado e impresora, y grabación y lectura de datos y programas a partir de cintas de casete o discos flexibles.

Nos hallamos, por tanto, en la primera generación de los sistemas operativos. Un primer paso importante ha sido la realización de intérpretes del lenguaje BASIC, por medio de los cuales se pueden transmitir comandos al ordenador. De este modo, el usuario no debe ya preocuparse de gestionar personalmente los canales de comunicación, programando las puertas de entrada/salida registro por registro. Es suficiente con una instrucción "PRINT", por ejemplo, para que el intérprete BASIC se ocupe de todas las operaciones de control necesarias.

Pero la verdadera revolución se produjo cuando, con el desarrollo tecnológico, fue posible comercializar unidades de disco y memorias a precios competitivos. Así se han empezado a crear los primeros sistemas operativos diseñados específicamente para los ordenadores personales. Hoy en día, por fin, el usuario dispone de medios muy eficaces para aprovechar al máximo, y con el grado de comodidad que desee su pequeño ordenador.

En los comienzos, como es lógico, cada firma constructora trataba de realizar un sistema operativo propio, de tal modo que fuera incompatible con cualquier otro ordenador. Esto hacía que fuera imposible utilizar un software de aplicación realizado para un ordenador distinto. Ahora bien, dado el enorme número de microordenadores presentes en el mercado, esto suponía una gran desventaja para los compradores, ya que cada aparato podía ejecutar únicamente los programas escritos específicamente para él;

la situación constituía también un problema para las pequeñas empresas constructoras, que no lograban desarrollar grandes paquetes de programas, pues resultaba ser una tarea demasiado grande para sus estructuras. La única salida viable era la de establecer un estándar, común para todos los equipos, para la realización del software.

En la segunda mitad de los años setenta empezó a difundirse en América un sistema operativo diseñado especialmente para microprocesadores de 8 bits: el CP/M. No es que sea precisamente el mejor sistema operativo bajo muchos puntos de vista: modo de gestionar los discos hace más bien lento el acceso, los mensajes de error son indescifrables y no es posible ninguna técnica refinada en cuestión de gestión de ficheros. Pero posee a cambio una gran ventaja: ha sido realizado de tal manera que puede ser adaptado con facilidad a todos los ordenadores que utilizan un microprocesador con código Intel 8080, Z 80 o compatibles, sin problemas de intercambio de memoria, periféricos y discos. De este modo se ha hecho realidad aquello que había sido siempre un sueño para la industria del ordenador personal: transportar el software de un equipo a otro.

La Digital Research, sociedad creadora del CP/M, ha realizado toda una gama de versiones de este sistema operativo, que se adaptan a casi todos los ordenadores personales en circulación, apropiándose nada menos que del 58% del mercado mundial (Fig. 6). Además, tras comprar la licencia, cualquier firma puede adoptar el CP/M a sus ordenadores con sólo seguir las instrucciones proporcionadas por la casa productora.

Mientras tanto, los dos colosos de la industria de los ordenadores personales no podrían quedarse de brazos cruzados. Así realizaron o encargaron sus propios sistemas operativos: el DOS de la firma Apple y el MS-DOS de MicroSoft (o PC-DOS) para IBM. Ambos han sido creados específicamente para determinados equipos, obteniendo un buen resultado. De todos modos también han adoptado, como sistema operativo adicional, el CP/M: la cantidad de software disponible para él (sobre todo juegos) es tal, que tanto Apple como IBM, para no perder un elevado número de posibles compradores, han tenido que aceptarlo.

El CP/M es ya un estándar, de hecho, para los ordenadores de 8 bits, y también tiene mucha difusión el CP/M 86 para los de 16 bits. Sin embargo, en el área de los procesadores de 16 bits la empresa IBM está imponiendo como estándar el MS-DOS: como vemos, de alguna manera se está repitiendo, a pequeña escala, la historia de la serie 360...

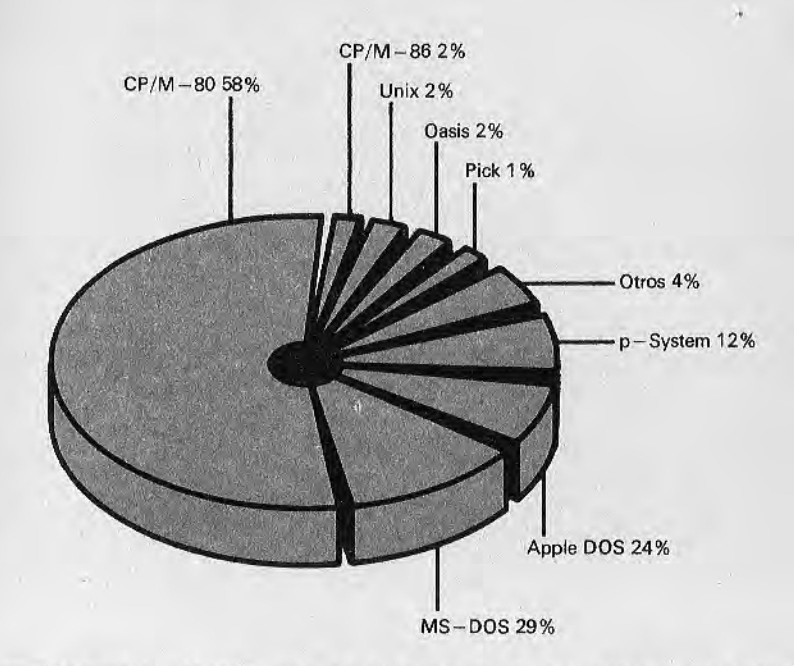


Figura 6.—Distribución en el mercado de los sistemas operativos más utilizados (datos de 1983).

Anatomía de un sistema operativo

Tras haber visto el nacimiento y desarrollo de los sistemas operativos tanto en ordenadores grandes como pequeños, veamos ahora su organización. Evidentemente, no se halla dentro de nuestros objetivos describir en este volumen la correspondiente a los grandes ordenadores, naturalmente mucho más complejos.

Observando la figura 7 puede llegar a tener cierta idea de la complejidad de las funciones de un sistema operativo. Cuando compramos un ordenador personal nos encontramos frente a toda una serie de dispositivos de cuyo interior sabemos poco o nada. Hay a nuestra disposición un teclado, una unidad de vídeo, una impresora, una unidad de discos y el ordenador en sí, compuesto a su vez por una unidad central (o CPU), la memoria interna y varios elementos de entrada/salida.

El sistema operativo debe encargarse de organizar todos estos dispositivos de manera que para nosotros puedan continuar siendo, en su interior, totalmente desconocidos. ¡Imagínese la cantidad de problemas que se nos plantearían si cada vez que nece-

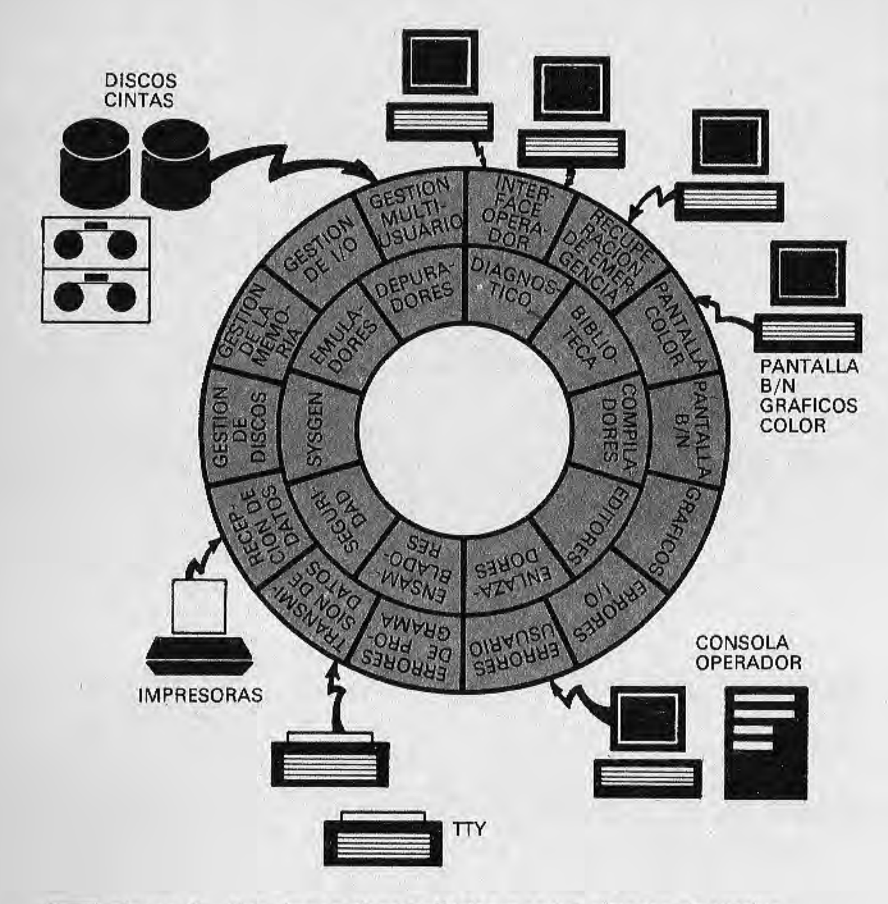


Figura 7.—Funciones desarrolladas por un sistema operativo.

sitáramos utilizarlos tuviéramos que ocuparnos de ellos! Vamos a poner, tan sólo como ejemplo, el funcionamiento de un terminal de vídeo. Cada vez que pulsamos una tecla y aparece en la pantalla el carácter correspondiente, el sistema operativo ha desarrollado toda una compleja serie de operaciones elementales: disponer una dirección de entrada de datos, codificar el carácter tecleado, tras haber eliminado las múltiples pulsaciones (debidas a rebotes mecánicos de las teclas), memorizarlo en un área interna, denominada "buffer", convertirlo en una forma visualizable y enviarlo a la pantalla. Como ha podido comprobar, se trata de una serie de operaciones demasiado complicadas y largas como para que estén todas las veces a cargo del usuario...

Pero el trabajo del sistema operativo no termina aquí. En efecto, una vez que el usuario ha completado el comando, éste ha de ser estudiado: un programa del sistema, denominado "analizador sintáctico" (en inglés "parser"), debe verificar si la sintaxis del comando es correta o no y decidir, por tanto, si debe hacer referencia a su memoria interna, a los discos o a otros dispositivos periféricos. En los próximos capítulos veremos cuáles son los grandes problemas que conllevan la utilización de estos recursos.

Pasemos ahora a examinar de qué manera se pone en funcionamiento un sistema operativo. Debido a sus dimensiones (una versión sencilla de CP/M ocupa fácilmente 8 Kbytes de memoria) normalmente se halla en disco. En el área de memoria EPROM reside un pequeño programa llamado "bootstrap" (de carga inicial) que, tras el encendido, toma del disco el sistema operativo, lo carga en memoria y le traspasa el control. Con esta operación se evita elegantemente imponer una elección definitiva del sistema operativo al ordenador: bastará cambiar la versión que se halla en el disco para dotar al ordenador de un sistema operativo renovado (incluso, a veces, distinto).

El CP/M, por ejemplo, ha sido modificado y puesto al día varias veces a lo largo de los años sin que los usuarios tuvieran que abrir para nada el ordenador; tan sólo es necesario cambiar el disquete y dejar que el bootstrap se preocupe de cargar la nueva versión.

Este método también se utiliza en los ordenadores más grandes: en el VAX 11/780, por ejemplo, es posible usar el VMS o el Unix indiferentemente, pues es suficiente con suministrar uno u otro sistema en el momento del booting (inicialización; es la ejecución del bootstrap).

En ciertos ordenadores domésticos y portátiles, en cambio, como el M10 Olivetti, el sistema operativo está grabado directamente en ROM: de este modo se evita la necesidad del disco, pero, por contra, queda vinculado al sistema operativo escogido.

Una vez cargado en memoria, el sistema operativo ofrece al usuario la posibilidad de elegir entre una variada gama de comandos y servicios. Los comandos forman, por regla general, parte del sistema y, por tanto, quedan todos cargados en RAM en el momento del bootstrap. Su ejecución será así muy rápida, bastando sólo con realizar el salto a una dirección de memoria determinada. Los comandos permiten llevar a cabo operaciones como el examen de la memoria del ordenador, la copia de ficheros (de archivos y de programas), el examen de los ficheros residentes en disco, o bien suministran información acerca de la longitud de un fichero, de la cantidad de memoria disponible o del espacio ocupado en el disco.

Otros elementos más sofisticados son los llamados "servicios"

(o utilidades-"utilities") suministrados por el sistema operativo, como editores, compiladores, ensambladores, editores y procesadores de texto y tratamiento de gráficos. Estos servicios no residen normalmente en memoria, dada la considerable cantidad de espacio que ocupan, sino que se transfieren únicamente cuando son solicitados por el usuario. De todas maneras hablaremos con más detalle de estos programas en el capítulo 3.

Panorámica de los sistemas operativos presentes en el mercado

Ahora que hemos tratado a grandes rasgos de su funcionamiento, vamos a ver cuáles son los sistemas operativos más difundidos hoy en día en el mundo de los ordenadores personales. En la figura 6 se muestra una visión del mercado actual, pero hay que tener en cuenta que se halla en constante y rapidísima evolución; varios de los que aparecen pueden disponer de más de un sistema operativo.

CP/M 80, el "bus de software"

Es el más difundido de los sistemas operativos para equipos basados en microprocesadores de 8 bits. Es modular, independiente del hardware y pensado para sistemas monousuario. Comercializado por Gary Kildall a comienzos de los setenta con la versión 1.4 al precio de 70 dólares, se ha desarrollado a lo largo de estos últimos años con varias versiones, de entre las cuales la 2.2 fue la más difundida.

Realizado por Digital Research, su éxito radica principalmente en dos factores: su reconfigurabilidad y la enorme cantidad de programas disponibles en el mercado. Desarrollado originariamente para los microprocesadores Intel 8080 y 8055 poseía el código ejecutable compatible con el Zilog Z-80. En versiones sucesivas ha sido adaptado completamente para el Z 80 que, al ser el procesador más comercializado, ha ejercido la misión de vehículo de arrastre del CP/M.

El CP/M se puede entender dividido en cuatro partes principales:

- BIOS: sistema básico de entrada/salida,
- BPOS: sistema operativo, básico de disco,
- CCP: procesador de comandos de consola,
- TPA: área de programa transitorio.

El factor que más le ha ayudado a imponerse ha sido su módulo BIOS, es decir, aquella parte que permite adaptar el CP/M a las características (memoria, periféricos y discos) de cualquier ordenador que utilice el Z 80. Todo esto sin olvidar que, según la opinión de la mayoría, el CP/M es un sistema operativo nada fácil de aprender por parte de los recién iniciados, con un pésimo diagnóstico de errores y un método de acceso a los discos bastante lento. Pero, aunque parezca curioso, es precisamente esta escasa optimización del sistema la que ha permitido una notable adaptación a las características de diferentes ordenadores. Con el CP/M se pueden gestionar programas de una longitud de hasta 8 Mbytes cada uno, que utilicen hasta 16 discos, con la posibilidad de proteger los programas individualmente y de repartir un disco en 16 áreas diferenciadas.

La última versión del CP/M es la 3.0, denominada vulgarmente CP/M Plus (disponible, por ejemplo, en el Amstrad 6128). Ofrece una organización de los datos en árbol (véase, más adelante, lo que comentamos en el Unix), una gestión de la memoria acorde con las grandes cantidades de ésta, disponibles hoy en día y sofisticaciones típicas de los sistemas operativos modernos.

Además, ha sido introducido en el mercado el CP/M Personal, totalmente residente en ROM, que permite implantarlo en ordenadores domésticos. Esto ha permitido también a los sistemas portátiles, como el PX-8 de Epson, disponer de un verdadero sistema operativo.

Apple DOS

Fue desarrollado especialmente para el Apple II, y gracias a la enorme difusión que éste ha tenido en el mercado ha obtenido un gran éxito. Posee un gran punto débil, que estriba en que es utilizable sólo en este ordenador.

Se trata de un sistema operativo estudiado a propósito para aprovechar las características del Apple, por lo que ocupa poca memoria y deja a disposición del usuario la mayor cantidad posible de RAM. Hay en el mercado una cantidad verdaderamente notable de software de aplicación. Así, puede encontrarse un número increíble de juegos (lo que sin duda alguna ha contribuido de forma determinante a la difusión de este ordenador en el mercado americano), pero también gran abundancia de programas de aplicación, desde los de administración de almacenes hasta otros más sofisticados que han hecho posible también la utilización del Apple en aplicaciones profesionales.

El futuro de este sistema operativo está claramente ligado a las intenciones de su empresa, que parece mostrarse decidida a

seguir utilizando este producto para sus futuras aplicaciones en el área de los 8 bits.

MS-DOS, el estándar de IBM

Desarrollado por la firma americana MicroSoft, se le conoce también como PC-DOS, ya que ha sido el sistema operativo elegido por IBM para su Personal Computer (IBM-PC). Tiene a su disposición una enorme cantidad de programas de aplicación, sobre todo en el campo de las aplicaciones profesionales. Fue escrito para los microprocesadores de 16 bits 8086 y 8088 de Intel, (el segundo tiene un bus de datos externos de 8 bits, pero internamente trabaja también con 16), de modo que funciona únicamente en equipos que adoptan este tipo de CPU.

Se ha iniciado una veloz carrera entre innumerables casas constructoras para realizar ordenadores de 16 bits basados en las CPU 8086/8088; son los llamados "compatibles" (IBM-PC, claro). El MS-DOS repite así el fenómeno CP/M en el área de los 16 bits. De todas formas, se trata de un sistema operativo avanzado en relación con sus predecesores de 8 bits, ya que está dotado de una estructura en árbol para catalogar los programas que se hallan en disco: ésta es precisamente una de las tendencias de los sistemas operativos de la última generación, como el Unix.

Bajo el punto de vista sistemático, el MS-DOS posee un gran mérito por la posibilidad de añadir con facilidad nuevas unidades de disco sin penosas operaciones. Bajo el punto de vista del usuario, en cambio, la mayor ventaja es la abundancia de software disponible. Al haberse impuesto ya como el más vendido entre los ordenadores personales, el IBM-PC ha inducido con rapidez a todas las empresas de software del mundo a comprometerse al máximo en la redacción de programas para este sistema, superando actualmente incluso a los disponibles para el CP/M.

Otra ventaja del MS-DOS es la del tratamiento de los mensajes de error que, mientras en el CP/M resultaba demasiado reducida y sibilina, es, en cambio, extensa y clara en este sistema.

Empujados por el éxito obtenido, se están produciendo hoy distintas versiones de este sistema, con mejores y nuevas prestaciones. Tal vez la más interesante sea la Windows, de la firma MicroSoft, que permite la ejecución de varios programas a un tiempo. Cada programa se "apropia" de una parte de la pantalla, de manera que el usuario puede observar su funcionamiento en paralelo. Sin embargo, no es posible todavía, en rigor, hablar de un verdadero sistema multitarea, como el Unix o el CP/M concurrente, ya que los programas, para poder funcionar en el modo que es definido como "concurrente", deben residir todos juntos

en memoria, lo que supone un fuerte límite por la cantidad de memoria disponible.

En los sistemas operativos más evolucionados, para remediar tal inconveniente se utiliza el disco de una manera muy intensa, tratando de hacer residir cada vez en la memoria central sólo pequeñas partes de los programas en ejecución, transportándolos del disco a la RAM cuando sea necesario y borrándolos cuando ya no hagan falta más.

La familia CP/M de 16 bits: muchos hijos, pero no todos afortunados

Con el advenimiento de los microsistemas de 16 bits, el CP/M no podía, por supuesto, quedar atrás. Así que fue reescrito en una versión para el microprocesador de 16 bits 8086 de Intel, llamada CP/M-86, que permite usarlo, por ejemplo, en el IBM-PC, que lo ha adoptado como alternativa a su MS-DOS.

Su filosofía es la misma que la del CP/M-80 (por lo que aún permanecen esos pequeños vicios de indescifrabilidad para el usuario), pero se han incluido ciertas mejoras que pretenden hacer del CP/M-86 uno de los victoriosos en la guerra desencadenada entre los sistemas operativos para los micros de 16 bits, aunque hasta ahora ha tenido poco éxito debido a la supremacía en el mercado del MS-DOS.

Pero la característica más interesante de los esfuerzos de Digital Research es el desarrollo de una familia CP/M de 16 bits: en efecto, hay disponible una variedad multiusuario, denominada MP/M-86, una versión multitarea-monousuario, el CP/M concurrente y el CP/M-68K para ordenadores basados en el microprocesador de 32 bits (y bus de datos externo de 16) 68000 Motorola.

El MP/M-86 (disponible también para 8 bits con el nombre de MP/M-2) es uno de los más conocidos entre los aún raros multiusuarios. Dispone de una gestión de la memoria a partición fija y ofrece también una forma simple, pero eficaz, en el blocaje de los registros.

El producto más interesante de toda la gama es, con toda seguridad, el CP/M concurrente. Este sistema permite a un solo usuario tener varios programas funcionando al mismo tiempo en un único ordenador. Puede, por ejemplo, estar corriendo un programa que realiza el balance mensual, mientras el usuario se dedica a escribir una carta en la pantalla y en una impresora está saliendo el listado o los resultados de otro programa. Este sistema operativo, que requiere al menos 256 kbytes de memoria y un abundante soporte de memoria de masa (normalmente un disco

rígido), ha sido realizado "ex profeso" para el IBM-PC, en el que funciona con regularidad, aunque ahora se halla también disponible para otros ordenadores que utilizan el 8086.

La versión con más miras hacia el futuro de todos los CP/M es, sin duda alguna, el CP/M-68K. Se trata, en efecto, del primero de toda la gama que está escrito en lenguaje C (véase más adelante el párrafo sobre el Unix); esto hace que, siempre que tenga a su disposición los compiladores apropiados, pueda ser utilizado en ordenadores con microprocesadores diferentes. La primera versión ha sido escrita para tratar de marginar al reciente Unix, que funciona precisamente, y con preferencia, en ordenadores basados en la CPU 68000 de Motorola. Pero este anhelo es muy difícil que llegue a cumplirse: el Unix ha llegado con fuerza y, con las características y prestaciones que ofrece, es con mucho superior.

El gran Unix

Considerado por todo el mundo como el sistema operativo de los ochenta, apareció en 1984 en el mundo de los ordenadores personales tras el acuerdo firmado entre IBM y Bell Labs para una futura implementación en el IBM-PC.

Es un sistema que permite el funcionamiento tanto multitarea como multiterminal, facilitando así la presencia de varios usuarios conectados en un mismo instante al mismo ordenador. El secreto del éxito de este sistema operativo, que ha obligado a rivales como el CP/M y el MS-DOS a inspirarse en él para mantener sus ventas, reside, por un lado, en haber sido escrito en C (lo cual le hace muy fácilmente transportable) y, por otro, en su filosofía, altamente innovadora.

Se pueden distinguir tres niveles: envolvente, núcleo y File System.

El envolvente (en inglés "Shell"), actúa como interface entre el sistema y el usuario. Para su manejo, el usuario puede utilizar un tipo de lenguaje a muy alto nivel, el "Command Language", que permite un uso bastante eficiente. En efecto, este modelo de lenguaje posee una forma particularmente coloquial que permite ser utilizado incluso por personas no expertas en el arte de la programación. Además, existe la posibilidad de escoger y personalizar este aspecto: cada firma que adopta el Unix puede remodelar el Shell; incluso la empresa madre permite al usuario la posibilidad de elegir entre dos.

El núcleo, en cambio, es la parte del sistema operativo que gestiona directamente el hardware del ordenador; se corresponde en la práctica con el BIOS del CP/M. Permanece en memoria

durante todo el tiempo en que el Unix está activo, controlando todas las funciones de acceso al disco, tratamiento de la memoria y operaciones de entrada/salida.

El File System (sistema de ficheros) es la parte más original. Está organizado con una estructura en árbol (Fig. 8) que permite, por ejemplo, la existencia de varios programas con un mismo nombre, aunque para ser alcanzados deberemos recorrer distintos caminos (pathway).

El usuario Unix tiene a su disposición tres posibles tipos de ficheros: ordinarios, directorios ("directory") y especiales.

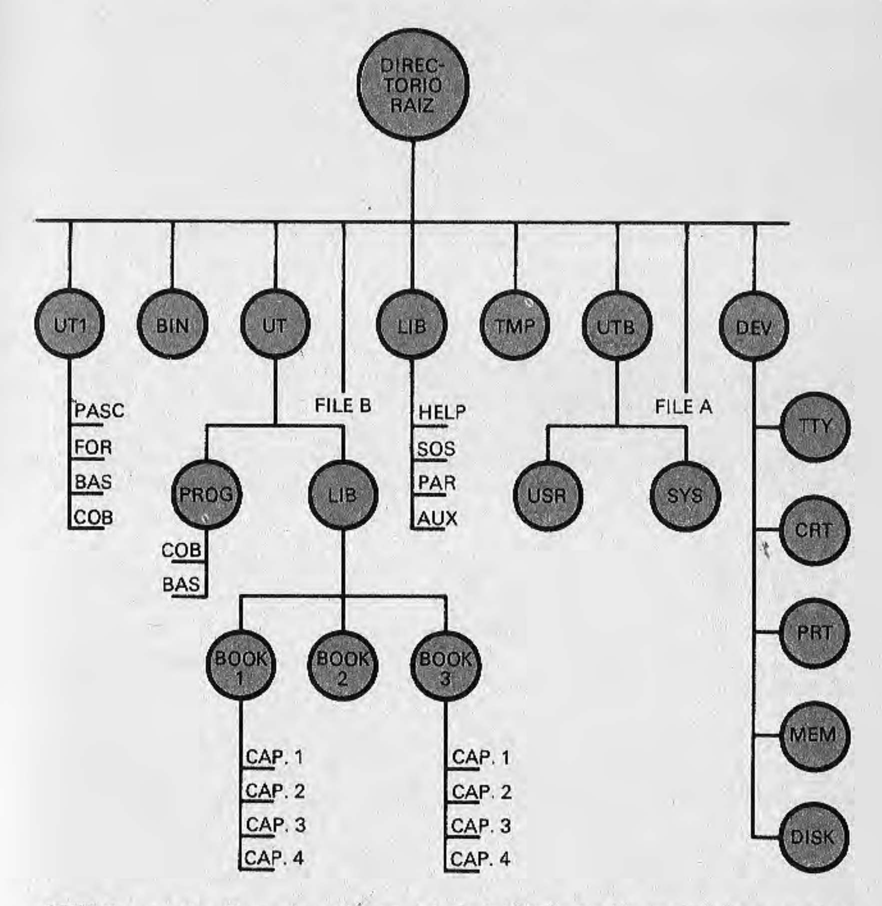


Figura 8.—Organización en árbol del Unix. La línea ondulada indica el recorrido, desde la raíz a los subdirectorios, necesario para alcanzar un fichero determinado.

Los ficheros-directorio son listas de otros ficheros o de otros directorios y se comportan como los ordinarios, pero no pueden ser escritos por otros programas; tan sólo puede manejarlos el sistema. Los directorios pueden ser varios y están todos dependientes del Directorio Raíz, que es un fichero que contiene los nombres de todos los Directorios. Recorriendo las sucesivas cadenas de Directorios se puede llegar a cada uno de los ficheros.

Anticipando algo sobre lo que insistiremos a su debido tiempo, el fichero CAP.2, incluido en el directorio BOOK, que forma a su vez parte de LIB, que, por último, se halla en la lista VT, es reconocido con el nombre compuesto VT/LIB/BOOK/CAP.2. En la figura 8 está remarcado con una línea el recorrido que, desde la raíz, lleva al fichero.

Los ficheros especiales son conceptualmente más difíciles de entender, al no ser una colección de datos o informaciones como los demás. En efecto, son utilizados para poder acceder a los canales de entrada/salida de una manera fácil y eficiente. Cada mecanismo de entrada/salida tiene al menos un fichero especial, cuyo nombre indica el tipo de dispositivo que gestiona. Un programa, o un usuario, que deseen utilizar, por ejemplo, una impresora, no necesitan saber nada acerca de este dispositivo: bastará que hagan referencia al fichero "impresora" normalmente para que el sistema operativo gestione la operación.

Otros sistemas operativos

Hay algunos sistemas operativos que en España no son aún muy conocidos, pero que en América tienen cierta difusión, a pesar de no ser en absoluto comparable a la de los sistemas tratados hasta ahora.

De éstos, el más extendido es el "p-System", desarrollado por la Universidad de San Diego, en California (la misma del UCSD Pascal). Se trata de un sistema para un solo usuario, cuya principal característica es la adaptabilidad a todos los ordenadores. Además, programas escritos con el p-System para microprocesadores de 8 bits, pueden ser fácilmente transferidos a un p-System de 16 bits. Al haber sido realizado en un ambiente universitario está muy difundido en América como instrumento didáctico, pero naturalmente escasean los programas de aplicación de tipo profesional. Este hecho, añadido a la falta de compiladores para lenguajes menos científicos que el Pascal y el FORTRAN, hace incierto su futuro.

Ha sido, en cambio, muy bien aceptado en el mundo comercial el sistema operativo Oasis. Apto para ordenadores de 8 y 16 bits, mono y multiusuario, es muy fácil de utilizar y ofrece buenas

prestaciones tanto bajo el punto de vista de la seguridad como de la organización de los ficheros. Por desgracia, debido a una serie de motivos, la versión de 16 bits ha entrado en el mercado demasiado tarde en relación con el Unix y ahora resulta muy difícil colmar la diferencia que se ha creado.

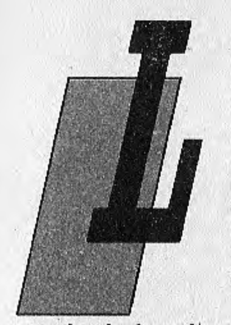
Totalmente orientado, sin embargo, hacia el mundo comercial está el BOS. Este es un sistema multiusuario, desarrollado en Inglaterra con el fin de soportar un lenguaje, el Micro Cobol, con unas características decididamente de gestión. Se trata de un sistema que funciona en la actualidad en varios ordenadores, permitiendo la gestión de los ficheros ISAM (Indexed Sequential Access Method). Al tener como único lenguaje el COBOL no permite asegurar la posibilidad de un gran desarrollo en el futuro.

Concluimos nuestra reseña con un sistema operativo muy interesante: el Pick. Desarrollado en un principio para los microordenadores ha sido ahora readaptado para los aparatos fundados en microprocesadores en versiones tanto monousuario como multiusuario. Ofrece prestaciones de gran relevancia, como la organización de la memoria virtual, y la capacidad (intrínseca) para gestionar bases de datos; es muy veloz y fácil de utilizar. En la actualidad está disponible sólo para unos pocos ordenadores basados en el microprocesador 68000 de Motorola, pero existe ya en América una versión para el IBM-PC.

CAPITULO II

LA MEMORIA CENTRAL: UNA ORGANIZACION COMPLEJA

Generalidades



La organización de la memoria principal, interna, central o primaria (en contraposición a la secundaria o de masa, de la que hablaremos en el capítulo 3), que de todas estas formas se puede llamar, es uno de los aspectos más importantes en el mundo del ordenador. Precisamente de entre las funciones más arduas para un sistema operativo destaca la gestión de la memoria interna, hasta tal punto que podemos afirmar que, junto con la de los discos, ha sido el factor de mayor influencia a la hora de proyectar y realizar los distintos sistemas operativos.

La memoria central es aquella parte del ordenador en que debe hallarse obligatoriamente un programa para poder ser ejecutado. Sin ella (y, por supuesto, el software necesario en cada ocasión) no sería posible ningún tipo de funcionamiento. Además, cuanta más memoria tengamos a nuestra disposición, más amplio será el uso que le podremos dar al ordenador, ya que nos permitirá escribir un mayor número de líneas de programa y ejecutar programas más complejos, que requieren más cantidad de memoria para tratar los datos.

En los ordenadores personales más económicos, en los que la memoria principal no es precisamente abundante, el sistema operativo proporciona mecanismos para utilizar la memoria secundaria, por ejemplo, una cinta magnética o un disco flexible, como expansión de la primaria.

Cuando el usuario ha agotado la cantidad de memoria disponible y no puede almacenar el programa que está tecleando, el

sistema operativo se encarga de grabar en la memoria auxiliar la información presente en la memoria principal, dejándola así dispuesta para aceptar nuevas informaciones. Este sistema, aunque sencillo, hace tremendamente lento el funcionamiento, sobre todo si la memoria principal disponible es pequeña. Por otra parte, al ser este dispositivo bastante costoso, el problema de la cantidad de memoria disponible resulta importante incluso en los grandes ordenadores, ¡donde hay nada menos que millones de bytes!

A causa de esta situación han sido desarrollados métodos que permiten, digámoslo así, "simular memoria" cuando no la hay; son las denominadas técnicas de memoria "virtual", que permiten una utilización bastante provechosa de la memoria disponible. Por el momento sólo están a disposición de los grandes ordenadores, pero serán con toda probabilidad utilizadas por los ordenadores personales de los próximos años, para permitir un mejor aprovechamiento de los mismos.

Organización de la memoria

A lo largo de la historia de los ordenadores el problema de la memoria interna ha sido tratado siempre con detenimiento, ya que puede incidir mucho en el precio del equipo. Como es fácil de suponer, la cantidad que había que incorporar se valoraba cuidadosamente tratando de instalar la menos posible, para no excederse en el coste, pero de forma que fuera suficiente para almacenar los programas del sistema operativo y los del usuario. Últimamente, gracias al gran desarrollo de la tecnología, los precios han bajado un poco, pero el problema, aunque menos agudo, sigue existiendo.

En años anteriores se pusieron a la venta ordenadores personales con una cantidad de memoria apenas suficiente para contener el más sencillo de entre los juegos de "marcianitos". De este modo el comprador se encontraba luego con la desagradable sorpresa que, al tratar de realizar un programa un poco complicado, éste no cabía en el ordenador...

Por otra parte, el problema de utilizar la menor cantidad posible de memoria sin disminuir las prestaciones del sistema ha sido un reto que, desde siempre, ha atraído a los estudiosos de los sistemas operativos, que han llegado a desarrollar soluciones verdaderamente ingeniosas. Gracias a ellos, como ya dijimos, es posible hoy en día hacer funcionar, en los grandes ordenadores, programas más grandes que las dimensiones reales de la memoria.

¿Cuál es la estrategia que el sistema operativo debe utilizar? ¿Será un modo de trabajo para un solo usuario, o para varias personas trabajando simultáneamente?

En el primer caso será posible elegir entre una forma de ubicación simple o múltiple, de forma que esté permitido un funcionamiento de tipo "multitask" (es decir, varios programas de un mismo usuario funcionando a un tiempo). En el segundo caso habrá que elegir la filosofía para la gestión multiusuario *"timesharing"*, multiprogramación con particiones de memoria o de petición de página.

Vamos a examinar con más detalle el funcionamiento de la memoria de un ordenador. Como ya hemos visto, hay dos tipos de memoria: la primaria, o central, y la secundaria, llamada también periférica o de masa. De esta última trataremos en el tercer capítulo del libro. La memoria central está generalmente constituida por circuitos de sólo lectura (ROM, PROM, etc.), y RAM, para lectura y escritura.

Los primeros, de sólo lectura, son a menudo de un tipo intermedio, como las EPROM (es decir, que pueden también ser reescritos por medio de aparatos específicos, bastante sencillos y económicos); son circuitos integrados que, programados por el fabricante en el momento de la construcción del ordenador, no pueden ser modificados por ningún programa (aunque, en casos como el antes citado de las EPROM, sí por el usuario). Son, por lo general, utilizados para contener programas de "bootstrap", pequeñas partes o, incluso, todo el sistema operativo (o un intérprete BASIC) en aparatos que no poseen discos.

La ventaja de estos dispositivos, además de la de no ser volátiles como la RAM, es que, al no poder ser reescritos desde el programa, garantizan una protección segura en contra de accesos no deseados. Además, no necesitan transferencias de información de los discos, lo que hace más veloz el funcionamiento de los programas contenidos en ellos.

El inconveniente es que, al no poder manipularse su contenido, conservan su campo de direccionabilidad de forma definitiva, mientras que a veces resulta útil reorganizar la disponibilidad de memoria en base a las exigencias del momento.

Las memorias de tipo RAM son empleadas, en cambio, para contener datos y programas durante el funcionamiento de éstos, ya que en esta fase es necesario que ambos se hallen en la memoria central, pues la CPU no puede acceder al disco cada vez que deba ejecutar una instrucción.

La memoria periférica (discos y cintas) es, pues, empleada sólo para conservar las informaciones que volverán a ser utilizadas. Cuando es necesario ejecutar un programa que reside en disco, se transfiere primero a la memoria central; el procesador lo ejecuta y (por regla general) el resultado es nuevamente traspasado

al disco (en otros casos los resultados se llevan sólo a impresora y/o pantalla). Cuando se apaga el ordenador, el contenido de la RAM es destruido, mientras que lo almacenado en el disco queda grabado de forma permanente.

En los ordenadores más modernos se utiliza también una memoria denominada "caché", que posee unas características parecidas a las de la RAM, pero con un tiempo de acceso mucho más breve. Estos dispositivos resultan muy caros, por lo que son utilizados con moderación. Cada vez que un programa es ejecutado, el sistema operativo transfiere la parte en ejecución (cuyas dimensiones dependen de la cantidad de memoria "caché" disponible) a estos circuitos ultra-veloces, donde el funcionamiento requiere tiempos mucho más cortos.

Naturalmente, la cantidad de esta memoria debe ser tal que evite que el tiempo empleado por el sistema operativo para transferir el programa desde la RAM normal a la veloz sea mayor que el ahorrado gracias a la "caché", pues en otro caso el sistema tendría efectos negativos en el tiempo de ejecución.

Para hacer aún más veloz la ejecución de un programa han sido estudiadas estrategias de búsqueda ("fetch") más refinadas. El "fetch" es aquella fase en que el procesador va a buscar en la memoria el código de la instrucción que debe ejecutar. Hasta ahora se había pensado que, al no ser posible conocer de antemano el código que el procesador iba a necesitar a continuación, no había más remedio que realizar esto conforme el procesador lo fuera requiriendo.

Hoy en día, sin embargo, se han desarrollado estrategias cada vez más sofisticadas que permiten, con cierta seguridad, prever dónde deberá ir el programa a buscar la siguiente instrucción, ahorrando así una gran cantidad de tiempo. En efecto, el pre-fetch se hace con antelación, en el preciso momento en que la CPU ejecuta la instrucción correspondiente. Queremos hacer notar que este método, parcialmente adoptado en la actualidad en algunas CPU de 8 y 16 bits, parece destinado a afirmarse en un futuro próximo. Se trata de técnicas muy actuales, incluso para los mejores ordenadores personales.

Cómo se gestiona la RAM

Los ordenadores personales, como los primeros que se crearon, utilizan una forma de gestionar la memoria del tipo "simple usuario con distribución contigua". Es decir, que sólo un usuario cada vez puede utilizar el ordenador y todos los recursos del equipo (CPU, memoria, discos, periféricos) están totalmente

a su disposición y, mientras no se utilizan, permanecerán en espera de órdenes.

Los programas se disponen en la memoria central según el esquema de la figura 1: en cabeza el sistema operativo con todos sus procedimientos, luego el programa del usuario y, por último, el área de memoria no necesaria, que permanece inutilizada. De este modo si el usuario desea ejecutar un programa más grande que la cantidad de memoria disponible, el sistema operativo le comunicará que esto es imposible.

La única solución sofisticada que este tipo de organización presenta es el uso de los "overlay": el programa del usuario es subdividido en rutinas, utilizadas a menudo, y secciones, que son utilizadas una sola vez; en la memoria central se cargan sólo aquellas partes del programa cuyo uso es requerido con más frecuencia. Las otras, en cambio, son cargadas y ejecutadas una por una, con un mayor empleo de tiempo, ciertamente, pero con la ventaja de poder trabajar sobre porciones de memoria más reducidas. Está claro que el esfuerzo de subdividir un programa es elevado, de modo que si el sistema operativo no es capaz de efectuarlo automáticamente, será preferible recurrir a esta técnica sólo en caso de absoluta necesidad.

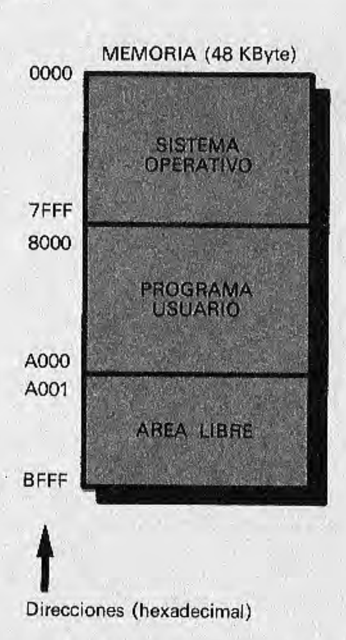


Figura 1.—Distribución simple contigua: típica partición funcional de la memoria.

El sistema de protección resulta mucho más fácil, pues al haber un solo usuario y un solo programa en funcionamiento, el único problema es impedir que el programa del usuario pueda ser escrito accidentalmente sobre las posiciones de RAM reservadas al sistema operativo. Para evitar esto, un registro específico se encarga de controlar todos los accesos ejecutados por el usuario. Así, cuando hay un intento de penetrar en la zona de memoria reservada al sistema operativo, se pone en marcha un mecanismo que interrumpe el programa y señala la infracción.

Reducir los tiempos muertos

En los años cincuenta se hicieron estudios estadísticos acerca del funcionamiento de un programa, y se cayó en la cuenta de que aproximadamente el 60-70% del tiempo de ejecución estaba en realidad dedicado a operaciones de entrada/salida, en las que el procesador no era utilizado. Así, desde entonces se trató de aprovechar al máximo estos tiempos muertos para mejorar las prestaciones del ordenador, dando origen a toda una serie de técnicas de gestión de la memoria que se comprometían a no dejar nunca ociosa la CPU.

Esta inquietud ha dado lugar a la multiprogramación: al separar completamente las fases de procesamiento de las de control de los periféricos es posible aprovechar mucho mejor el procesador. Para hacer esto es necesario realizar la gestión de entrada/salida de una forma "inteligente", de tal modo que sea posible el manejo de los periféricos de una forma totalmente desvinculada del procesador central. Así se podrá, por ejemplo, efectuar la impresión de un programa, al tiempo que se está ejecutando otro.

De hecho, ya hay en el mercado algunos ordenadores personales dotados de dos procesadores: uno que es la CPU propiamente dicha y otro que se encarga del control de los periféricos. En los ordenadores más grandes se implementan unos "controladores de canal" que se ocupan de gobernar, de forma autónoma, los dispositivos externos. Surge entonces el problema de cómo optimizar el uso de la memoria en estas condiciones.

Observando la figura 2 se puede ver cómo, cuando el programa 1 libera la atención de la CPU para ejecutar operaciones de entrada/salida, el sistema operativo indica la ejecución del programa 2. Podría ocurrir que éste tuviera que hacer también operaciones de I/O antes de que el programa 1 hubiera terminado las suyas. En este caso, el sistema operativo podrá decidir la ejecución de un tercer programa, y así todas las veces que sea necesario hasta que el primero esté disponible para volver a utilizar la CPU.

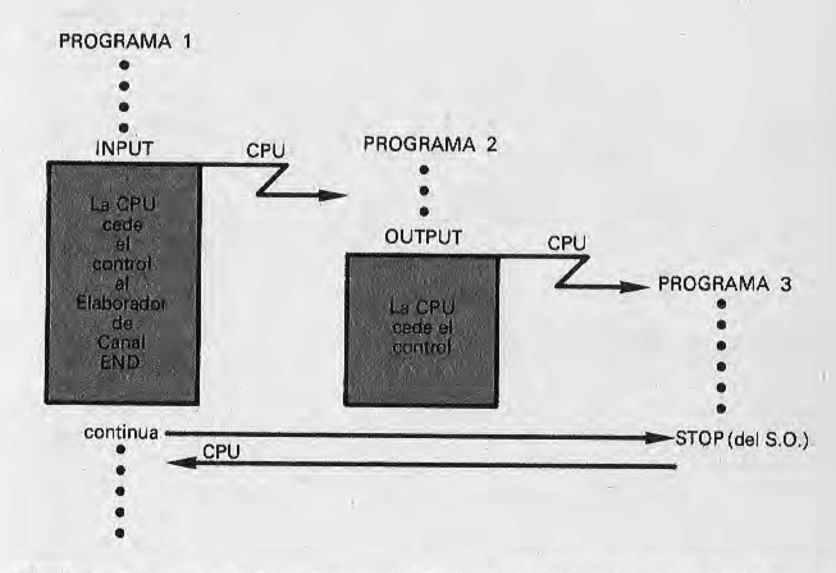
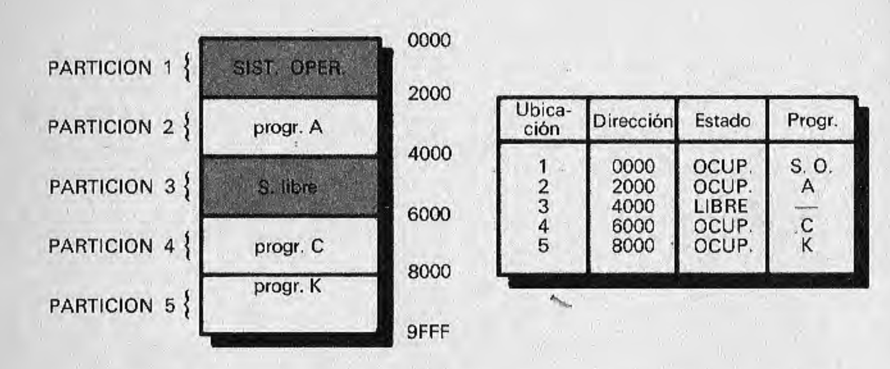


Figura 2.—Multiprogramación: paso de la atención de la CPU de un programa a otro en las fases de entrada/salida.

Naturalmente, para pasar de un programa a otro es necesario salvar cada vez las informaciones relativas al que dejamos en el instante en que se suspende su ejecución y cargar las del nuevo cuya ejecución retomamos. Este procedimiento resta un poco de tiempo, llamado tiempo de servicio, a la CPU. Mientras el "tiempo de servicio" que estos procedimientos de intercambio comportan sean similares a los tiempos muertos del procesador, el asunto marcha bien. Pero cuando estos tiempos no son suficientes se asiste a un progresivo deterioro de las prestaciones, que puede llegar a provocar incluso un "System Overhead" (desbordamiento del sistema), durante el cual el equipo gastará todo su tiempo en ejecutar programas de servicio en lugar de los programas de usuario.

Tras comprobar que este tipo de gestión comporta una utilización mejor del ordenador, debemos considerar de qué maneras repartir los programas en la memoria. Multiprogramación significa, naturalmente, mantener varios programas a un tiempo en la memoria central: se trata de hacerlo con la mayor eficacia posible. El método más sencillo es el de subdividir la memoria en "particiones" y situar un programa en cada partición según el esquema de la figura 3. En una tabla guardada en la memoria se refleja la numeración de las particiones, las dimensiones de cada una, sus



Ubicación	Dirección	Estado	Progr.
1	0000	OCUP.	S. O.
2	2000	OCUP.	A
3	4000	LIBRE	—
4	6000	OCUP.	C
5	8000	OCUP.	K

Figura 3.—Distribución en particiones fijas: el derroche de memoria puede ser muy grande.

direcciones y si están libres u ocupadas. Su asignación puede llevarse a cabo de forma estática o dinámica.

En el primer caso (reflejado en la figura 3) las particiones son fijas, por lo que se produce un notable derroche de memoria, ya que sólo en contadas ocasiones los programas poseen las dimensiones que se habían calculado y, por tanto, abundante espacio permanecerá inutilizado.

En la asignación dinámica las áreas son asignadas en base a las dimensiones del programa que se haya cargado. De esta manera la memoria está mejor aprovechada, pero también se produce un fenómeno negativo: a fuerza de subdividir la memoria ocurre que algunas pequeñas zonas no serán aprovechadas, dando lugar a la llamada "fragmentación". Para organizar lo mejor posible la distribución de los programas en la memoria o, para decirlo en lenguaje técnico, ubicar y reubicar las particiones, han sido desarrollados ingeniosísimos algoritmos: los más famosos son los llamados "first fit" y "best fit".

En el "first fit" hay una lista de las zonas libres dispuesta según el orden creciente de sus direcciones de partida, que son examinadas una a una hasta hallar la primera que satisfaga las exigencias de espacio del nuevo programa del usuario. De este modo permanece libre el fondo de la memoria para satisfacer posibles exigencias de grandes trabajos.

El *"best fit"*, en cambio, utiliza una tabla con las dimensiones de los espacios libres, ordenadas por orden creciente de su tamaño. Esto permite la utilización del área que se adapte mejor al programa que hay que guardar, sin tener que buscar a lo largo de toda la tabla.

Un estudio realizado por IBM utilizando un IBM 360/65 ha dado los siguientes resultados: con programas de medianas dimensiones (128 kbytes), el porcentaje medio de tiempo de espera del procesador con una distribución sencilla contigua era del 65%; con un sistema de multiprogramación con cuatro programas en funcionamiento, este porcentaje se reducía aproximadamente al 10%.

Ventajas de la memoria virtual

Un método para tratar de resolver el problema de la fragmentación es el de compactar periódicamente las zonas usadas de la memoria. La compactación es un método de reubicación dinámica que consiste en llevar todos los segmentos utilizados a un extremo de la memoria, combinando todos los espacios libres en el otro extremo. En la figura 4, en (a), se ve la memoria ocupada por tres procesos (A, B y C) antes de la compactación, y en (b) observamos cómo queda la distribución de memoria tras realizar esta operación.

Esto conduce, sin embargo, a una serie de problemas posteriores, como la modificación de todas las instrucciones que hacen referencia a una dirección concreta de la memoria y de todas las

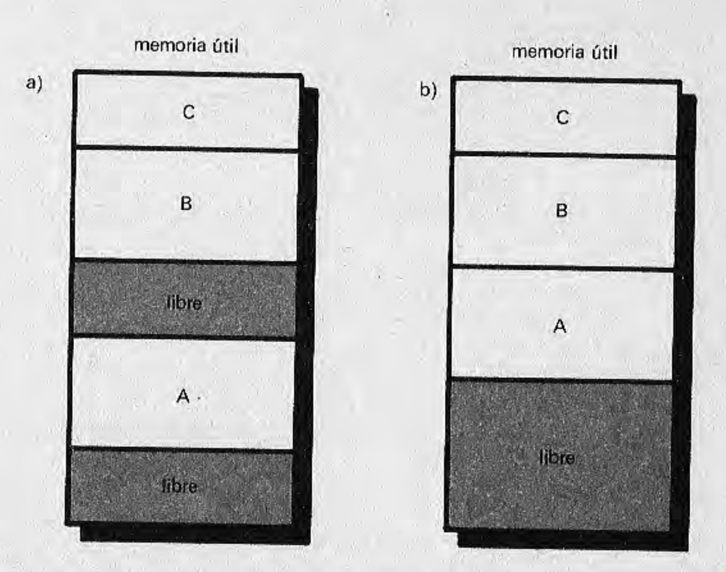


Figura 4.—Asignación y disponibilidad de memoria antes (a) y después (b) de un proceso de compactación.

estructuras de datos que hacen uso de punteros a direcciones. Es decir, para poder "compactar" el programa debe ser "reubicable": las direcciones especificadas por el programador en la definición de las instrucciones deberán poder ser transformadas en direcciones físicas diferentes en cada desplazamiento de los programas efectuados por la compactación.

Esta es la técnica básica necesaria para utilizar las llamadas "Memorias Virtuales". En los sistemas que hacen uso de este tipo de organización, la dirección de una posición de memoria usada en las instrucciones de un programa, codificado y listo para la ejecución, puede diferir de la dirección (física) efectiva en que escribirá su contenido el sistema operativo. Este hecho provoca una distinción entre la dirección escrita por el programador, que llamaremos "dirección virtual", y la que será en realidad utilizada durante la ejecución, que llamaremos "dirección física".

El modo de actuar no es nada sencillo, pues deberán ser cambiadas tanto las direcciones de las distintas instrucciones, como los operandos que hacen referencia a la memoria (escritura y lectura, saltos, llamadas). Aún más difíciles de resolver son los direccionamientos indirectos, es decir, aquellos en los que la dirección efectiva está a su vez contenida en una celda de memoria: en este caso, la operación de reubicación deberá ser aplicada dos veces.

Una posible solución viene descrita en la figura 5. En ella se hace uso de un registro base para la reubicación. Cada operación de direccionamiento es ejecutada sumando a la dirección obtenida por la instrucción el contenido del registro base. Así, en el ejemplo de la figura, el programador desea cargar en el registro A el byte contenido en la dirección "virtual" 10000. El sistema operativo sumará a esta dirección el contenido del registro base, en este caso igual a 30000. El dato entonces será leído en la dirección 40000, donde habrá sido reubicado.

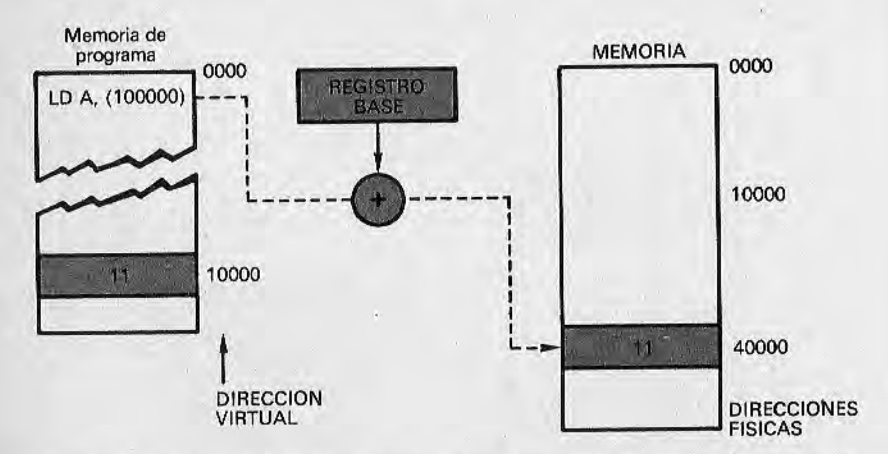


Figura 5.—La distinción entre direcciones virtuales y físicas permite la reubicación de la memoria. El registro-base convierte la dirección virtual en física.

La ventaja de este método es poder gestionar de un modo más flexible la memoria, eliminando la fragmentación de la que hablamos anteriormente. El tiempo requerido para efectuar las operaciones de reubicación y el precio de los registros especiales de reubicación disminuyen las ventajas de este método.

Organización en páginas

Una primera mejora se produce con la organización en páginas. Con este método el espacio de memoria utilizado por el programa es dividido en "páginas" de longitud (pequeña) prefijada. La memoria física, en cambio, es subdividida en bloques, cuyas dimensiones son idénticas a las de las páginas. De este modo es posible colocar las páginas del programa en bloques de memoria físicamente no contiguos.

Este afinamiento permite aprovechar mejor las zonas libres de la memoria colocando, por ejemplo, sólo algunas de las páginas de un programa. La correspondencia entre páginas y bloques es realizada en base a un conjunto de tablas, llamadas PMT (Page Map Table, tablas de mapa de páginas). Naturalmente es el sistema operativo el que se encarga de manipular y poner al día estas PMT.

En la figura 6 puede verse un ejemplo, con cuatro programas colocados a un tiempo en memoria. El programa B ha sido subdividido en cuatro páginas que, por medio de la PMT, son posicionadas en cuatro bloques de la memoria física.

La gestión de esta técnica es bastante compleja y requiere un gran trabajo por parte del ordenador. Las conversiones páginas-bloques son efectuadas por registros muy veloces, para acelerar así los tiempos de ejecución, pero por estas características resultan muy caros.

El desarrollo de este método lleva a uno de los más utilizados hoy en día en la gestión de la memoria en los grandes sistemas multiusuarios: la "paginación".

La característica común a todos los sistemas tratados hasta ahora era que permitían cargar en memoria los trabajos de varios usuarios a un tiempo, hasta la saturación de la memoria física disponible. Una vez agotada, no se podía poner en funcionamiento

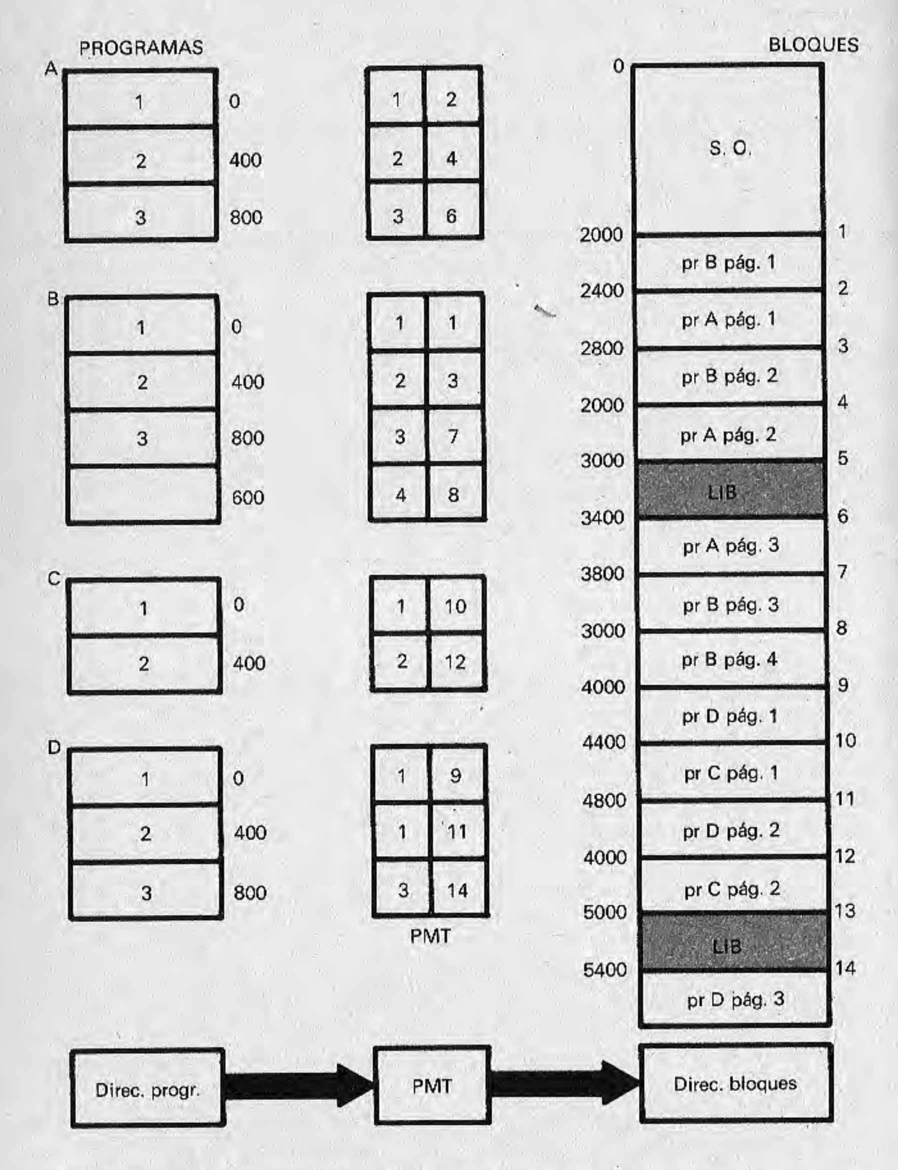


Figura 6.—Gestión de la memoria por páginas. Las páginas que componen cada programa son colocadas en la memoria en bloques no contiguos. El sistema operativo se encarga de situar las páginas en los bloques según vayan quedándose libres. Para esto se sirve de la PMT (Page Map Table). Por ejemplo, las páginas 1, 2, 3, 4 del programa se hallan colocadas, respectivamente, en los bloques 1, 3, 7, 8 de la memoria física.

un nuevo trabajo hasta haber liberado la memoria lo suficiente como para poder cargarlo.

Al observar la ejecución de un programa subdividido en páginas queda de manifiesto que no es necesario que todas las páginas estén cargadas simultáneamente en memoria para asegurar su funcionamiento, ya que las rutinas de un programa no son utilizadas todas seguidas. Algo parecido ocurre con los archivos de datos, por lo general muy extensos: sólo una parte es utilizada cada vez durante las distintas fases de la ejecución. De modo que es posible realizar un sistema operativo que permita gestionar y aprovechar una "memoria virtual" cuyas dimensiones sean bastante mayores que las (internas) físicas disponibles.

Este tipo de organización implica, naturalmente, incrementar la complejidad del sistema operativo, que deberá decidir qué páginas tendrá que mantener en la memoria y cuáles no y, además, las operaciones que será necesario llevar a cabo cuando un programa requiera el acceso a una página que no se halla en memoria en ese momento. En este último caso, si no queda espacio disponible habrá que quitar algunas páginas para hacer sitio a las otras, necesarias; el problema estriba en saber cuáles quitar, ya que si retira unas páginas no adecuadas, que el programa va a solicitar después, todo el proceso anterior habrá de ser repetido, con la pérdida de tiempo que esto supone.

Para determinar de qué páginas se puede prescindir se recurre a teorías estadísticas que operan según una filosofía de tipo "predictivo" sobre la futura utilización de las páginas, basándose en la historia pasada del programa. De este modo, los algoritmos de tipo "FIFO" (First Input First Output, primero en entrar, primero en salir) eliminan la página que fue cargada en memoria hace más tiempo, mientras que los del tipo "LRU" (Least Recently Used, usada hace más tiempo) sustituyen aquella a la que no se hace referencia desde hace más tiempo. Ambos métodos poseen ventajas y defectos, por lo que no es posible predecir si uno es mejor que el otro.

El principal problema de la paginación es que, dada la cantidad de funciones necesarias, la sobrecarga del sistema operativo es notable, por lo que se hace más lenta la velocidad del ordenador. Este u otros modos de funcionamiento de parecida complejidad pueden, llegando al límite, conducir al fenómeno denominado "thrashing" (pérdida del control) en el que se consume casi todo el tiempo en ejecutar los programas de gestión, permitiendo al usuario disponer sólo del 1% del tiempo disponible. En esencia, el thrashing consiste en que el procesador solicita páginas a mayor velocidad de la que se le pueden suministrar. Llega un momento en que el procesador tiene que detenerse ("procesador ocioso"), pues todos los procesos están esperando a que se produzca una transferencia de página.

Otra técnica de gestión de la memoria es la segmentación. Es conceptualmente parecida a la paginación, pero utiliza en lugar de páginas "segmentos" de dimensiones variables, que son definidos por el usuario en el interior de la instrucción como parte de la dirección. Si se utiliza la segmentación, toda dirección se compondrá de un identificador de segmento y un desplazamiento. Así, una instrucción Ensamblador del tipo:

LD RO, <2> 1000 H

significará: carga en el registro RO el contenido de la celda de memoria 1000 (hexadecimal) perteneciente al segmento 2. Naturalmente, el sistema operativo procederá a operar una conversión segmento-bloque para colocar el programa según sus exigencias. Lo interesante de este tipo de organización reside en que los microprocesadores de 16 bits permiten implementar esta técnica con facilidad. Parece lógico, por tanto, suponer que los futuros desarrollos de los ordenadores personales estarán orientados según esta filosofía.

Un caso típico: la memoria del IBM-PC

La memoria central del IBM-PC está subdividida, como todas, en ROM y RAM. La ROM, programada por la empresa, contiene aquella parte del sistema operativo que no puede ser modificada. Para efectuar cualquier cambio habrá que sustituir el circuito integrado por otro.

Los programas más importantes contenidos en la ROM son las rutinas de gestión de los dispositivos de entrada/salida y el intérprete BASIC para cintas. En la versión base de este ordenador la ROM ocupa 40 Kbytes, situados al fondo del área de memoria direccionable. Hay también disponibles zócalos para otros 8 Kbytes por si el usuario quiere utilizarlos para almacenar programas de una forma permanente.

Para cargar los programas que normalmente residen en disco, como el Advanced Basic Interpreter, y los del usuario hay disponibles además 64 Kbytes de memoria RAM. En la System Board (placa del sistema) hay también espacio para otros 192 Kbytes de RAM (hasta 256), que el usuario puede incorporar. Para necesidades mayores se pueden añadir otras placas que permiten un máximo de hasta 640 Kbytes.

El espacio de memoria está organizado sobre 1 Mega-byte (1.048.575 bytes), según el gráfico de la figura 7, partiendo de la dirección 00000 hasta la dirección FFFFF (hexadecimal). Para acceder a ella pueden crearse bloques, llamados segmentos, de un

DIRECCION		FUNCION	
DECIMAL	HEXADEC.		
0 · · 48 K	00000 · · 0C000	16÷64K Byte RAM System Board	Proporcionada con el ordenador
64 K · · 624 K	10000 · · 9C000	hasta 448K instalables por el usuario	Para usos generales
640K	A0000	Reservada	
656K ... 688K	A4000 ... AC000		Para visualizaciones y usos particulares
704K	B0000	VIDEO B/N	
720K	B4000		
736K	B8000	VIDEO Color	
752K 768K	BC000 C0000		
· · 944K	EC000	192KB - expansión de memoria	Aplicaciones especiales
960K	F0000	Reservada	
978K · · 1008K	F4000 · · FC000	48KB - ROM del Sistema Operativo	Proporcionados con el ordenador

Figura 7.—Distribución de la memoria en el IBM-PC.

tamaño de 64 Kbytes cada uno, que pueden empezar en cualquier punto de la memoria, con la única condición de que su dirección de partida sea divisible por 16. Esto equivale a que en la numeración hexadecimal la última cifra de la derecha de cualquier dirección inicial de segmento sea siempre 0.

La parte baja de la RAM, desde la dirección 0 en adelante, se utiliza para cargar el sistema operativo DOS y el Advanced Basic desde el disco, además de los programas que el usuario desee utilizar. Por tanto, los caracteres escritos mediante el programa Editor tendrán que ser introducidos en esta zona de memoria.

Un centenar de Kbytes en el centro de la memoria están reservados para la gestión de la pantalla, en blanco y negro, en color e incluso gráfica, además de para otras tarjetas que es posible introducir en el ordenador.

En el fondo, en cambio, están colocados los 48 Kbytes de ROM reservados para el sistema operativo. El modo de direccionamiento utilizado es del tipo "segmentado", es decir, cada celda de memoria está localizada por un par de números, como vimos anteriormente: uno que identifica el segmento y el otro la posición relativa de la información en su interior (offset). De este modo, la forma de direccionar una posición de memoria ya no es unívoca. En efecto, vamos a suponer que tenemos que leer el contenido de la celdilla situada en la dirección física 28519: si el segmento al que nos estamos refiriendo parte de la dirección 0, el offset será precisamente el 28519. Si, en cambio, comienza en la localización 15000, la dirección a la que tendremos que referirnos será la 13519 (todo ello en decimal).

Para acceder a segmentos externos al utilizado normalmente por el intérprete BASIC es necesario utilizar la instrucción DEF SEG. En efecto, al especificar un valor comprendido entre 0 y 65535, éste es multiplicado por 16 y se convierte en la dirección inicial del segmento en uso desde ese punto del programa en adelante. Así, la instrucción:

```
10 DEF SEG = &HC000
```

fijará el inicio del segmento en la dirección $C000_H \times 16_D = C0000_H$. Las instrucciones sucesivas harán referencia a este número para calcular las direcciones.

El BASIC utilizado en el IBM-PC permite también acceder directamente a las celdas de memoria por medio de una serie de instrucciones que podrán hacer referencia a una dirección comprendida en el segmento activado por la DEF SEG o a una dirección simbólica, expresada por un nombre. Para leer un byte de la memoria se utiliza la instrucción PEEK (n), donde "n" expresa una dirección comprendida entre 0 y 65535. Así, al poner, por ejemplo,

```
10 CELDA = PEEK (&H5000)
```

el programa leerá el contenido de la dirección 5000_H y lo meterá en la variable CELDA.

En cambio, para escribir se utiliza la instrucción POKE n,m que introduce el valor "m" (un byte) en la dirección de memoria "n". La instrucción

```
POKE 1000, 10
```

escribirá el número 10_D en la celda de memoria de dirección 1000_D en el segmento en curso.

Para transferir un bloque de memoria al disco (o cinta) o leer un programa del disco (o cinta) y ponerlo en una posición determinada de memoria existen las instrucciones BSAVE y BLOAD. Con la instrucción

```
BSAVE "COPIA", &H1000, &H100
```

el sistema operativo cogerá los 100_H (en hexadecimal = 256 en decimal) bytes que parten de la dirección 1000_H del segmento activo y los grabará en el disco, asignándoles el nombre de COPIA. Por contra

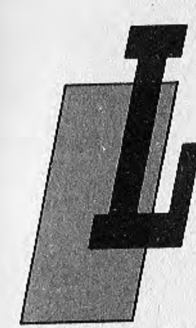
```
BLOAD "COPIA", &H100
```

leerá del disco el contenido del programa COPIA poniéndolo en la memoria central, en el interior del segmento corriente, a partir de la posición relativa 100_H.

En caso de que ésta no estuviera definida, el programa será cargado a partir del punto indicado por una variable del programa situada en memoria. La instrucción VARPTR proporciona la dirección, en el interior del segmento activo, de la variable solicitada. Es oportuno recordar también que el espacio de memoria ocupado por esta variable dependerá de su tipo: un número entero ocupará dos bytes, mientras que un carácter ocupará un solo byte.

CAPITULO III

SISTEMAS DE ALMACENAMIENTO DE DATOS: ORDEN E INTEGRIDAD



Los programas en ejecución en un ordenador intercambian información con los usuarios a través del teclado, terminal de vídeo, impresora, lectores de tarjetas, etc. Estos programas tendrán, por tanto, que almacenar la información recibida de una manera estable y permanente. Con este fin habrá que utilizar un dispositivo, accesible en el menor tiempo posible, que pueda almacenar una cantidad considerable de datos y que sea accesible para todos los usuarios del sistema, pero de forma que los datos no puedan mezclarse los unos con los otros. Los dispositivos de almacenamiento masivo utilizados en la actualidad son los discos, rígidos y flexibles, y las cintas magnéticas.

En este capítulo examinaremos las características físicas de discos y cintas, los modelos existentes y su organización. Trataremos el concepto de fichero, la manera en que se estructuran los datos que hay que almacenar en disco y la parte de sistema operativo que se encarga de la gestión de los ficheros.

Memoria de masa

Bajo el nombre de memoria de masa, o memoria secundaria, están englobadas las estructuras que permiten almacenar grandes cantidades de datos de una forma permanente, es decir, discos y cintas magnéticas. Podemos hacer distinciones entre estos dispositivos en base a características como precio, cantidad de información almacenable, velocidad y tipo de acceso. Estos parámetros van unidos a sus características físicas y, por desgracia, a

menudo se hallan en conflicto. Así no es posible tener un soporte velocísimo, capaz de almacenar enormes cantidades de datos y a un precio irrisorio. En los ordenadores personales (como en los micros domésticos actuales) había hasta hace poco como única memoria auxiliar el casete de cintas: lento, poco fiable, con escasa capacidad de almacenamiento, pero de bajo coste. Naturalmente, este último era el parámetro predominante. Ahora, en cambio, se utilizan discos flexibles y rígidos de dimensiones reducidas, realizados expresamente para un mercado en fuerte expansión.

Cintas magnéticas

La memoria de masa más económica es, indudablemente, la cinta magnética. Forman parte de esta categoría tanto las cintas corrientes utilizadas en los ordenadores personales como las grandes de 2400 pies usadas en los grandes centros de cálculo.

La capacidad de almacenamiento de una cinta magnética se mide por:

- longitud física de la cinta, que se suele expresar en pies,
- densidad de grabación, que representa la cantidad de caracteres que se pueden almacenar en la unidad de longitud. Los códigos más frecuentemente utilizados son: ASCII y EBCDIC.

La conexión entre el ordenador y el casete se realiza por medio de un dispositivo de interface que se encarga de transformar la información disponible en el ordenador en formato binario en otra de tipo analógico válida para realizar la grabación (Fig. 1). La señal analógica puede estar modulada en amplitud o en frecuencia (Fig. 2).

En la modulación por amplitud se utilizan técnicas de tipo PAM (Pulse Amplitude Modulation) que hacen corresponder al valor 1 del bit una señal de mayor amplitud que la asociada para el valor 0. Esta técnica permite una implementación muy económica, pero ofrece también menos garantías, ya que los ordenadores personales que adaptan esta solución presentan, por regla general, problemas ligados a la intensidad sonora de grabación, ruidos y otros problemas de este tipo.

Una mejora notable se obtiene al emplear técnicas de modulación de frecuencia, tipo FSK (Frequency Shift Key). Utilizan una correspondencia bit-frecuencia, haciendo corresponder una frecuencia diferente a cada valor del bit. Pero este sistema tampoco se libra de problemas, resintiéndose de fenómenos como el Wow (fluctuación de baja frecuencia) y el Flutter (oscilación), típicos de

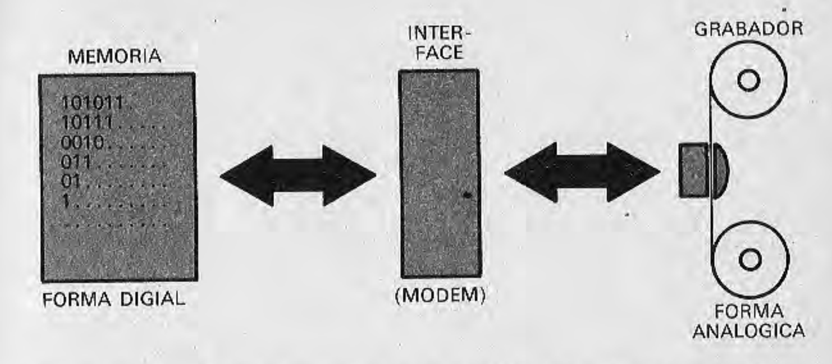


Figura 1.—Transferencia de información del ordenador al casete.

los casetes de cintas. De todos modos, el grado de confianza en estos aparatos está, en realidad, en función del grado de complejidad de la estructura previamente elegida y, por tanto, del precio.

Las ventajas de estos dispositivos al compararlos con los discos son, sobre todo, dos: lo económicos que resultan y la gran cantidad de información que es posible almacenar. Por esto precisamente es por lo que las cintas son utilizadas, por regla general, para el "back up" (copia de seguridad) en ordenadores medianos y grandes. Es decir, son empleadas para almacenar, al final de la jornada o de la semana, todos los programas y datos residentes en disco para crear un archivo de seguridad.

El principal inconveniente del almacenamiento en cinta reside en la escasa velocidad que presenta este sistema. Esto se debe principalmente a la anchura de banda de la cinta y el dispositivo de conversión de los datos (que no permiten sobrepasar una cierta velocidad en la lectura/escritura) y, sobre todo, al hecho de que la cinta es un dispositivo de acceso (físico) secuencial. Para entender esto bastará pensar en una cinta musical corriente: si nosotros queremos, por ejemplo, escuchar la sexta canción, tendremos que avanzar la cinta hasta este punto, pasando por las canciones anteriores aunque no nos interesen. Lo mismo ocurre con los ordenadores: para tener acceso al enésimo fichero habrá que perder cierto tiempo haciendo correr la cinta hasta el punto deseado. En cintas grandes este tiempo desaprovechado puede ser enorme (incluso de bastantes minutos).

Para solucionar al menos en parte este problema han sido estudiados mecanismos de búsqueda más veloces. Volviendo al ejemplo anterior, si nosotros sabemos exactamente dónde se halla la canción que hemos elegido, no tendremos que escuchar toda

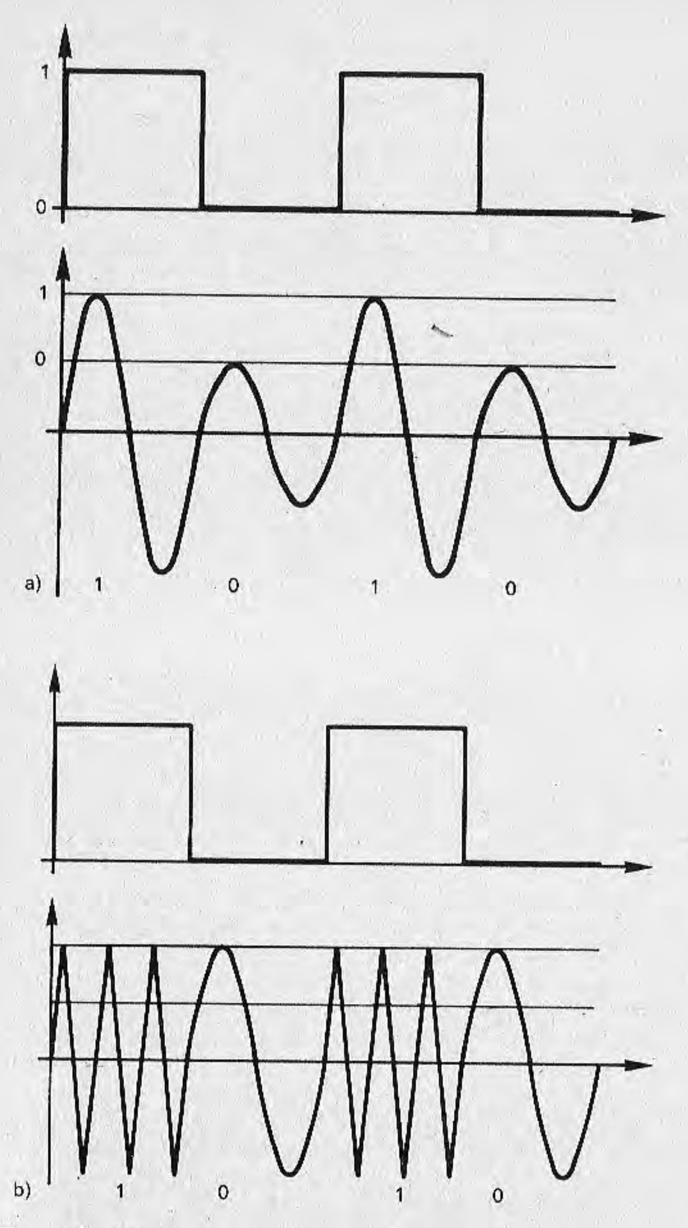


Figura 2.—Técnicas de modulación. a) modulación en amplitud (más sensible a los ruidos); b) modulación en frecuencia del tipo FSK: los valores 1 y 0 del bit están representados por señales (analógicas) de distintas frecuencias.

la cinta hasta encontrarla, sino que iremos con la máxima velocidad posible al número de vuelta determinado previamente.

De este modo, si al comienzo de la cinta ponemos un índice de todos los ficheros que contiene y que nos dé el número de vuelta en que se encuentran, se reducirá notablemente el tiempo de acceso.

Hay que tomar la precaución, cada vez que se añade un fichero a la cinta, de poner al día el índice que se halla al comienzo. Esto implica que habrá que dejar un espacio libre, en el cual el sistema operativo se encargará de grabar y, en cada momento, actualizar el índice de los ficheros, cuyo número estará, por tanto, limitado por la cantidad de espacio que dejemos al comienzo de la cinta. Aunque a pesar de estos métodos las cintas no podrán competir nunca en velocidad con dispositivos de acceso directo como son los discos, sí mejoramos su eficiencia.

Discos rígidos y flexibles

Volviendo a utilizar la analogía musical, consideremos esta vez un disco. Como es bien sabido, para oír una parte determinada del mismo basta con situar la cabeza lectora en el surco adecuado; efectuamos un "acceso directo" a la canción elegida. El disco de un ordenador funciona de la misma manera, sólo varía su aspecto. Está realizado sobre un soporte duro o flexible y el surco, visible en el disco musical, es aquí invisible, ya que se trata de una traza magnética, llamada "pista". La pista, a su vez, está dividida en "sectores", con capacidad normalmente para 128 bytes.

Los discos flexibles, disquetes o floppy disks (Fig. 3) son los más utilizados en los Ordenadores Personales y micros por su considerable velocidad de acceso, fiabilidad y precio,
nable. Construido sobre un soporte de plástico flexible recubierto por un material magnético, el disquete se divide, a su vez, en varias clases, en base a diferentes parámetros.

En relación con las dimensiones tenemos hoy en día tres clases:

- el clásico floppy, con un diámetro de 8 pulgadas;
- el minifloppy, cuyo diámetro es de 5,25 pulgadas;
- y el microfloppy, con un diámetro de 3,5 pulgadas.

Hay además disquetes de tamaño no estándar, algunos de dimensiones muy reducidas, y otros con un sistema de grabación de las pistas en espiral en vez de con círculos concéntricos, que están destinados a aplicaciones particulares.

Hubo un tiempo en que la diferencia en las dimensiones im-

plicaba una mayor o menor cantidad de información almacenable. En la actualidad, gracias a los desarrollos tecnológicos, todo esto ha sido modificado: en efecto, al aumentar la densidad de grabación de datos, un minifloppy puede contener incluso un millón de bytes. Vamos a proceder ahora a otras clasificaciones.

La primera es en relación con la superficie de grabación: se distinguen sencilla o doble (por una o por ambas caras). Esta, más que una distinción entre los discos en sí, se refiere a las unidades de disco (es decir, las unidades de I/O), ya que son éstas precisamente las que pueden o no leer las dos caras del disco. Cualquier disquete puede ser utilizado por ambas caras, pues están las dos magnetizadas; como mucho, será preciso hacer las perforaciones utilizadas por la unidad de disco para controlar la protección del soporte.

En cuanto a la densidad de grabación, tanto las informaciones contenidas en la pista como el número de pistas presentes en el disco pueden compactarse más. La densidad de grabación de la pista sencilla puede ser doble y cuádruple, mientras que el número de pistas de un disco puede ser duplicado.

La última distinción se realiza en relación con el modo de identificar los sectores: pueden ser "soft" o "hard sectored". Los "soft sectored", que son la casi totalidad de los discos utilizados en los ordenadores personales, tienen un agujero que se corresponde con el comienzo de las pistas: un dispositivo fotoeléctrico se encarga de identificar el primer sector de la pista, mientras que los sucesivos serán identificados por un mecanismo "software" que conoce el número y situación de los sectores que hay en la pista. Los "hard sectored", en cambio, poseen un orificio para cada sector, por lo que la identificación es puramente mecánica.

La fiabilidad que ofrecen estos dispositivos es hoy considerable, a pesar de que, sobre todo en relación con los disquetes de elevada densidad de grabación, los discos rígidos son preferibles para aplicaciones en que se requiere gran confianza acerca de su seguridad.

Los discos rígidos, normalmente de tecnología Whinchester, son en la actualidad el medio más veloz, capaz y fiable disponible como memoria de masa por los grandes ordenadores.

Gracias al desarrollo de la tecnología estamos asistiendo a la realización de discos rígidos con dimensiones como las de mini y microfloppys, con características iguales a las de sus hermanas mayores, pero a un precio netamente inferior.

La producción de estos dispositivos está, por el momento, orientada hacia el mundo de los ordenadores profesionales y personales más evolucionados, que requieren un gran soporte de memoria de masa.

Veamos ahora las características de los discos rígidos. Pueden dividirse en removibles y fijos, y éstos, a su vez, en sencillos y múltiples. Construidos sobre soportes metálicos (de ahí la denominación de "rígidos" poseen dimensiones iguales o, más a menudo, superiores a las de los disquetes (14 pulgadas). La distinción entre discos removibles y fijos indica que los primeros se pueden extraer de la unidad de disco (como los floppy), por lo que con una sola unidad de disco se pueden manejar varios discos. Los sistemas de discos rígidos utilizados por los grandes ordenadores están formados por pilas de discos rígidos con varias cabezas.

Organización de una memoria de masa

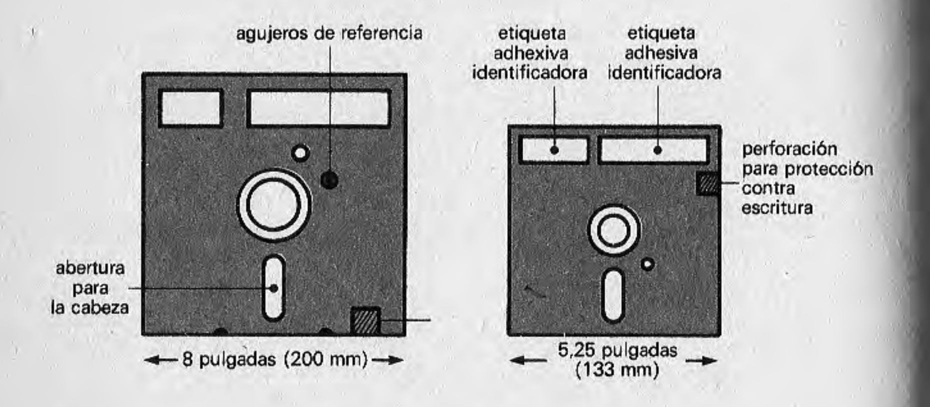
Como hemos visto en el apartado precedente, un disco está dividido en pistas y sectores (Fig. 3). Una pista es una línea de circunferencia que, al girar el disco a gran velocidad, pasa rápidamente debajo de la cabeza, pudiendo ésta leer o escribir sobre la pista.

Los discos estándar están, en general organizados con 48 pistas por pulgada en los discos de densidad sencilla, y con 96 pistas por pulgada en los de doble densidad. Esto se traduce en una media de 40 pistas disponibles para un minifloppy de densidad sencilla y 80 para uno de doble densidad, en cada cara. La pista, a su vez, está dividida en sectores, cuyas dimensiones varían, pero, en general, van desde una capacidad mínima de 128 bytes a una máxima de 4096 bytes.

En los minifloppy de densidad sencilla cada pista está dividida en 16 sectores de 128 bytes cada uno; tendremos, por tanto, 2 kbytes de datos por pista. En un disquete de 8 pulgadas, como el Shugart SA 800, pueden almacenarse 5,2 Kbytes por pista, para un total de 400 Kbytes por cara. De todas maneras, al utilizar un minifloppy con doble densidad y doble cara pueden alcanzarse los 370 Kbytes de un IBM-PC, por ejemplo, comparables a los 400 Kbytes de un floppy disk corriente de 8 pulgadas.

Estas cifras aumentan considerablemente al tratarse de un disco rígido: un Starfire de 10 Mbytes posee 306 pistas, con 32 sectores por pista.

Los discos nuevos se venden vírgenes o, como decimos en nuestra jerga, "no formateados". Esto se debe al hecho de que muchas firmas organizan sus propios discos de una manera totalmente diferente, por lo que uno grabado bajo el sistema operativo CP/M no podrá ser leído con el sistema operativo MS-DOS, y viceversa. La operación de formatear es aquella con la que el sistema operativo inicializa un disco.



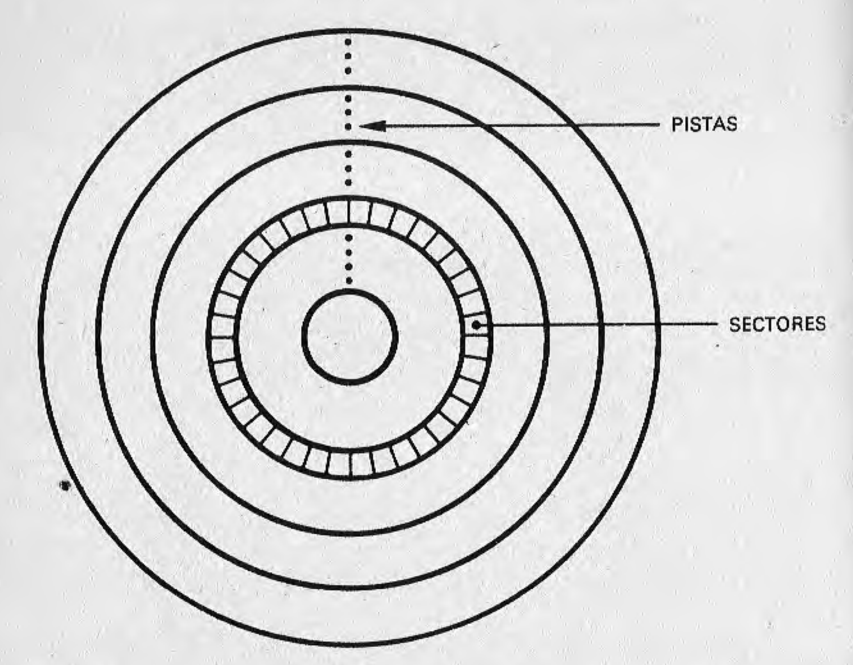


Figura 3.—Disquetes (floppy disk) de 8 pulgadas y minifloppy de 5 1/4 pulgadas. Subdivisión en pistas y sectores.

El comando "Format" se encarga (en los discos soft-sectored) de dividir el disco en pistas y sectores, grabando las informaciones necesarias para el movimiento de la cabeza en la fase de búsqueda del dato, además de la creación del directorio.

La distribución inicial de los sectores en el interior de la pista comporta algún problema en la organización; no conviene ordenarlos de forma secuencial, contiguos. Veamos, en efecto, qué ocurre durante el acceso a dos sectores consecutivos. Tras haber leído el primer sector el sistema operativo se dispone a repetir las mismas operaciones para leer el segundo, pero mientras tanto el disco ha seguido girando (con una velocidad que puede ser del orden de 3600 rpm), por lo que, para acceder al siguiente sector, la cabeza deberá esperar a que se haya completado el giro. El tiempo que se pierde en esta operación puede llegar a los 40 y 100 milisegundos, según la velocidad del disco utilizado.

La mayor parte de los sistemas operativos no utilizan un sistema de ordenación de los sectores de tipo secuencial, sino una técnica llamada de "interposición", en la que los sectores están colocados según la secuencia de la figura 4. De este modo, a la hora de acceder al sector sucesivo, el sistema operativo tendrá tiempo de reaccionar, sin tener que esperar al cumplimiento del giro.

El principal problema de la gestión de los discos estriba en la manera de reservar espacio para los archivos de datos y los de programas, organizados en bloques denominados registros ("re-

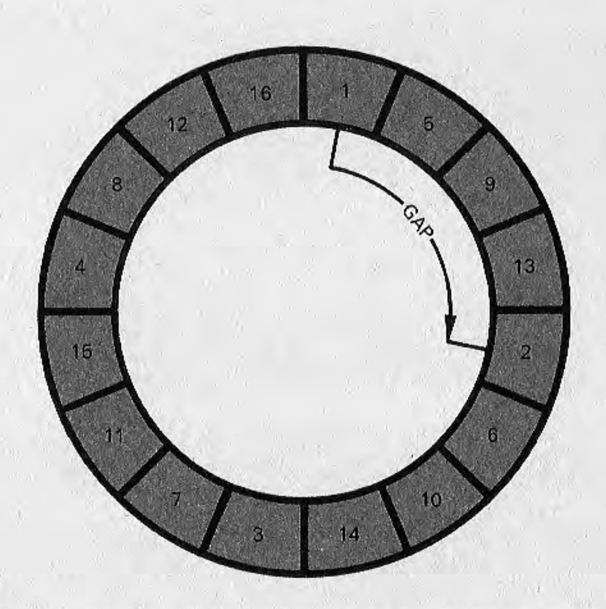


Figura 4.—Técnica de interposición de los sectores en las pistas del disco: el gap da al sistema operativo tiempo para disponerse a leer el siguiente sector.

cord") que son, en cierto modo, el equivalente de las páginas en la gestión de la memoria central. Es preciso optimizar el espacio disponible y reorganizar el disco al ser borrado o grabado un programa. Podría ocurrir que el espacio recuperado no correspondiera con el necesario para colocar un nuevo programa. En este caso, es obvio que no se deberá utilizar una colocación secuencial, es decir, la grabación de un registro tras otro, en orden numérico para cada fichero, sino un sistema algo más sofisticado.

Una de las técnicas más utilizadas es la de los punteros adelante-atrás (Fig. 5). Cada registro posee dos informaciones suplementarias, una situada en la cabeza y otra en la cola, que dan, respectivamente, la posición (pista-sector) del registro precedente y la del siguiente. Así, el sistema operativo no tiene problemas a la hora de reconstruir, cuando tenga que cargarlo en la memoria, un programa fragmentado en innumerables registros esparcidos por todo el disco.

Otro método consiste en confeccionar una lista con todos los registros en que está subdividido el fichero, de manera que el sistema operativo podrá acceder directamente al interior del programa; pero esta ventaja conlleva los inconvenientes del espacio ocupado por la tabla y el tiempo necesario para realizar la conversión tabla-registro.

Sea cual sea el método usado, el hecho de borrar un programa determina el nacimiento de áreas vacías, de dimensiones variables, que el sistema operativo debe reorganizar para almacenar nuevos ficheros. Esta operación de limpieza se denomina "garbage collection" (literalmente "recogida de basura") y puede hacerse de noche, cuando el ordenador está más libre, o bien cuando el sistema operativo no encuentra ya suficientes sectores para colocar el fichero necesario.

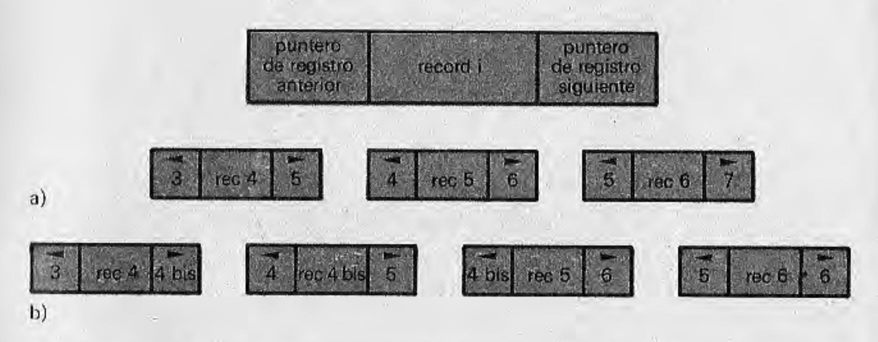


Figura 5.—Registro con punteros adelante-atrás: a) *organización;* b) *ejemplo de inserción de un nuevo registro.*

En realidad, esta función es bastante compleja y no siempre da los resultados deseados, por lo que generalmente se suele preferir reescribir el disco tras haberlo reformateado.

Otra técnica muy sencilla y rápida para informar acerca del estado de los sectores (libres u ocupados) es la utilización de un "bit map" (en la figura 12 se observa uno). Se trata de un mapa de los sectores del disco donde cada bit corresponde a un sector: el primer bit, al primer sector, y el último bit, al último sector. El estado del bit corresponde al estado del sector: si el bit está a 1, el sector está libre, lo contrario si está a 0.

La última operación efectuada por el Format consiste en reservar cierto número de sectores del disco para un índice general, denominado Catálogo o, más comúnmente, Directorio, que contiene todas las informaciones relativas a los ficheros que residen en el disco.

Veamos ahora brevemente de qué manera está organizado el formateado de una cinta. Por sus características físicas la cinta permite sólo accesos secuenciales, por lo que los ficheros estarán organizados simplemente según bloques contiguos con, en todo caso, un encabezamiento especial para distinguir bloques de datos de bloques de programas. Por desgracia, cada firma ha adoptado un método diferente para organizar las cintas (encabezamiento de los bloques, longitud, etc.), por lo que es prácticamente imposible leer con nuestro ordenador cintas grabadas por el de una casa diferente.

Ficheros: esos desconocidos

Con el nombre genérico de fichero entendemos hoy toda secuencia de informaciones grabadas en un soporte físico. Esta secuencia es utilizada para agrupar bloques de datos, programas de usuario o de sistema, es decir, que recoge un grupo de informaciones homogéneas, como pueden ser el conjunto de las instrucciones de un programa, el código ejecutable, una base de datos para un archivo, etc.

Como vimos anteriormente, la unidad más pequeña de información a la que podemos hacer referencia en un disco es el sector: su formato, elegido en el momento de inicializar el disco, puede ser de 128 bytes o un múltiplo de éste. Los ficheros se subdividen en registros y los registros en campos (Fig. 6). Esta es la mínima unidad de información a la que puede hacer referencia el usuario cuando quiera acceder a un fichero. El número de bytes que componen un registro puede algunas veces ser especificado por el usuario según sus necesidades. Posteriormente, el progra-

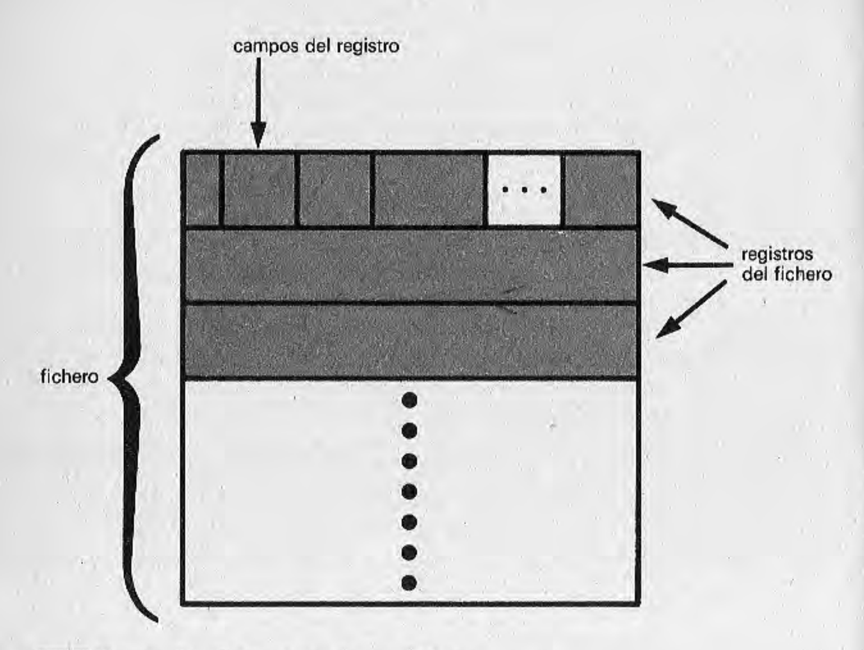


Figura 6.—Estructura de un fichero.

ma de gestión del disco tendrá que efectuar la transformación registro-sector.

En el interior del disco el fichero está catalogado en un índice que contiene una serie de informaciones suplementarias denominadas atributos. Estos atributos suelen ser:

1. propietario del fichero;
2. dimensiones;
3. posición en disco;
4. dirección en memoria;
5. fecha de creación;
6. fecha de utilización más reciente;
7. tipo;
8. protecciones;
9. formato del registro.

A continuación detallamos el significado de estos atributos:

1. El propietario (o creador) del fichero indica, en los sistemas multiusuario, el poseedor o el origen del fichero.

2. Este parámetro es presentado al usuario en forma de número de registros, ya que ésta es la unidad de información que él ve. Pero habrá que tener en cuenta también otro parámetro: el formato del registro que, al ser definible por el usuario, puede variar de un fichero a otro. Sólo si se considera la unión de estos datos podremos saber las dimensiones reales de un fichero.

3. La posición en disco indica la pista y el sector en que comienza realmente el fichero en el disco.

4. La dirección en memoria es, en cambio, un dato definido por el usuario, que sirve para dar una dirección absoluta a los procedimientos ejecutables (llamados corrientemente programas). En efecto, es el programa de gestión de los ficheros el que, por regla general, decide en qué zona de la memoria cargar el fichero retirado del disco, pero el usuario puede asignar al fichero una locación de memoria absoluta en la que el programa será cargado y ejecutado.

5. La fecha de creación (en ciertos sistemas se indica hasta la hora) informa acerca de cuándo ha sido creado el fichero. Este dato es útil cuando se utilizan nombres parecidos para revisiones de ficheros y ya no nos acordamos de qué versión se trata.

6. La fecha de utilización más reciente indica, en cambio, cuándo fue la última vez que se modificó o consultó el fichero en cuestión. Se apreciará la utilidad de esta información al aportar sucesivas modificaciones al mismo fichero. Los sistemas operativos más evolucionados se encargan de poner al día automáticamente la versión más reciente del fichero, creando otro nuevo (con un sufijo numérico que indica la cronología) cada vez que se accede a él en una fase de "editing" (edición).

7. El tipo es un parámetro que especifica el formato del fichero, es decir, la forma en que ha sido estructurado. Los tipos pueden ser:

- fuente (.SOU),
- de texto (.TXT),
- código objeto (.OBJ),
- código ejecutable (.EXE),
- listado del programa (.LIS),
- o expresan también el lenguaje en que ha sido escrito el programa (.PAS, .FOR, .BAS, .ASM, etc.).

8. Con la evolución de los ordenadores y la creciente importancia de los datos en ellos guardados y de los programas ha sido necesario desarrollar formas de protección para los ficheros que impidan que puedan ser leídos, borrados, reescritos e incluso el conocimiento de su existencia. Es posible proteger un fichero de forma que otro usuario que tenga la posibilidad de acceder al mis-

mo disco no pueda manipular o investigar las informaciones en él contenidas. En los ordenadores más grandes, utilizados por varios usuarios a un tiempo, han sido adoptadas técnicas de llaves de acceso, subdividiendo a los usuarios en cuatro grupos:

- dueño, quien ha creado y es propietario del fichero;
- grupo, usuarios que comparten con el dueño cierto número de recursos del fichero;
- mundo, todos los demás;
- sistema, el Sytem Manager, que debe poder acceder a todos los ficheros.

En ciertos sistemas monousuarios y en los ordenadores personales pueden protegerse también el fichero y todo el disco de la escritura. Esto sirve para defender los ficheros de nuestras propias distracciones.

El File System

Uno de los componentes más importantes del sistema operativo es el programa que gestiona los ficheros, denominado File System. Sus principales funciones son:

- gestión del acceso: proporciona el acceso a los datos contenidos en el fichero en base a su organización;
- gestión de los ficheros: organiza el almacenamiento de los ficheros en disco, las referencias, controla el uso de los ficheros comunes y garantiza su seguridad;
- asignación del espacio en disco: se encarga de ceder el espacio disponible en el disco a los ficheros;
- verificación (check) de los ficheros: es una de las funciones más delicadas, pues es la que verifica si los datos contenidos en los ficheros son correctos, encargándose de señalar si alguna anomalía ha restado fiabilidad a la información contenida en un determinado sector del disco.

El contenido de esta parte del sistema operativo, que de por sí es compleja en un ordenador personal con un solo disquete, es tremendo cuando se trata de gestionar un gran centro de cálculo. Basta pensar que aquí hay decenas (o centenares) de usuarios, cada uno de los cuales posee centenares de ficheros; hay en estos casos, por lo general, varias decenas de discos disponibles y ciertas órdenes de organización jerárquica que regulan la posibilidad de acceso de los usuarios a los recursos que comparten.

El File System tiene, por tanto, que clasificar y atender un número enorme de peticiones de acceso a discos, garantizando, al mismo tiempo, la integridad de las informaciones y el acceso en un tiempo tal que el usuario no se dé cuenta de su existencia.

En la figura 7 podemos ver una estructura típica en árbol, sobre la que el File System organiza a los usuarios y sus ficheros correspondientes. El índice raíz corresponde al índice general del disco; los índices usuarios corresponden a los directorios de los distintos usuarios, por último, las "hojas" terminales son los ficheros de los usuarios, contenidos en sus directorios.

Estructuras de este tipo y aún más complejas son ya de uso corriente en los grandes sistemas multiusuarios, como el Unix, en que la enorme ramificación de usuarios y ficheros hace necesaria una estructura particularmente compleja que permita una administración eficiente del sistema.

El File System debe ser capaz de ejecutar cierto número de operaciones, como:

- crear, modificar y borrar un fichero;
- permitir que, en un sistema multiusuario, un fichero pueda ser utilizado por varios usuarios;
- intercambiar informaciones entre diferentes ficheros;
- posibilidad de reclamar un fichero, en sistemas de varios discos, sin tener que especificar en qué dispositivo físico se halla;
- visualizar la información contenida en los ficheros independientemente de la estructura que hayan adoptado al ser almacenadas en disco (registro, sectores, etc.), de forma que el usuario puede acceder a ellas con la mayor comodidad posible;

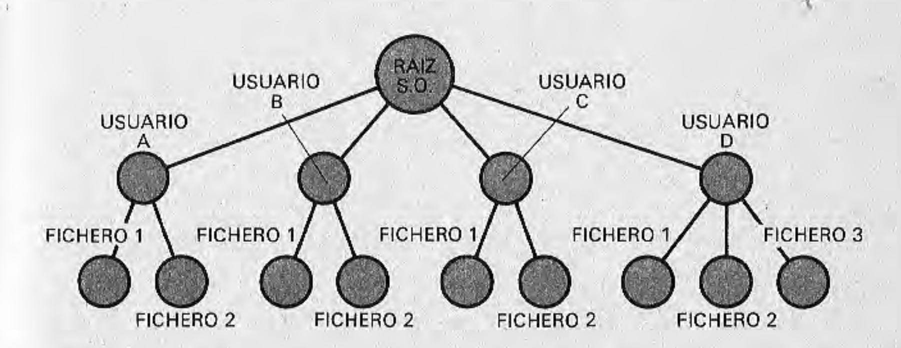


Figura 7.—Estructura en árbol de los ficheros. Permite una clasificación mejor y ofrece la posibilidad de dar el mismo nombre a ficheros que pertenecen a subdirectorios diferentes.

— crear un sistema de "back up" automático que permita al usuario, si por cualquier motivo pierde el fichero sobre el que está trabajando, recuperar al menos parte de su labor. Esta función debe ser realizada automáticamente en cada sesión de edición: cuando el usuario accede a un fichero que ya existe, se hace una copia del fichero que permanecerá de reserva en el disco.

Podría suceder que, por exigencias de un programa, fuera necesario acceder a algunos datos del fichero contenido en disco sin tener por esto que ejecutar una fase de edición. Pues bien, el File System proporciona al usuario una serie de alternativas al sistema con las que es posible hacer varias operaciones en los ficheros:

— CREATE: crea un nuevo fichero, encargándose de incluir su nombre en el índice del disco y de reservarle un espacio;
— DELETE: destruye un fichero ya existente, eliminando su nombre del índice y dejando libre el espacio reservado para él;
— OPEN: prepara un fichero para el acceso por parte de un usuario;
— CLOSE: concluye el acceso al fichero, impidiendo cualquier otro acceso hasta el siguiente OPEN;
— RENAME: cambia el nombre del fichero, actuando a nivel del índice del disco;
— COPY: duplica el fichero, cambiando el nombre si se desea;
— APPEND: permite la unión de dos o más ficheros en uno solo;
— SET PROPERTIES: permite la modificación de las características del fichero que se hallan en el índice del disco;
— READ/WRITE: permiten leer o escribir, respectivamente, los sectores del fichero en disco directamente.

Uno de los accidentes que ocurren con mayor frecuencia, sobre todo con los floppy disks, es que se pierde la conexión entre un sector y otro, lo que determina que el usuario no pueda ya leer el fichero. Para estos casos hay unos programas, denominados Disk Repair, que inspeccionan todos los sectores del disco, eliminando los alterados y reestableciendo la conexión entre los que están en buen estado, lo que permite al usuario recuperar la mayor parte (ese al menos es el propósito) de la información.

Bases de Datos: jerarquías y relaciones entre los datos

El desarrollo tecnológico y la demanda cada vez mayor ha hecho posible que los dispositivos de memoria de masa, como los discos rígidos y flexibles, hayan adquirido unos precios asequibles. Esto ha empujado a los usuarios a crear ficheros de datos más grandes, almacenados permanentemente en disco. Las aplicaciones comerciales e industriales que requieren grandes archivos para la gestión de contabilidad, tratamiento de personal, etc., han hecho necesaria la creación de una estructura de datos muy amplia y compleja que ofrezca garantías y seguridad. Así nacieron las Bases de Datos (o Data Base), que añaden a estas ventajas otras muy interesantes.

Una organización empresarial posee muchas aplicaciones que utilizan los mismos datos; por ejemplo: un programa de gestión de salarios y otro de organización del personal para la repartición del trabajo harán uso de una base de datos común que contenga los empleados de la empresa. Si estos programas trabajaran de forma totalmente independiente habría un derroche inútil de memoria, ya que los datos estarían duplicados; será mucho mejor organizar una sola lista de empleados a la que puedan acceder ambos programas.

Ya hemos visto una de las peculiaridades que una base de datos ofrece: la posibilidad de ser compartida por varios procesos. Pero, naturalmente, surge en paralelo el problema de la seguridad de las informaciones hechas "públicas". Es necesario organizar la estructura con distintos privilegios de acceso, de manera que cada usuario esté autorizado sólo a ciertas operaciones. Habrá así quien podrá, por ejemplo, modificar el importe de un salario, quien tan sólo podrá leerlo o quien, tal vez, podrá leer sólo el de ciertas personas. Por otra parte, el hecho de compartir la información también tiene sus ventajas: al tener que trabajar sobre la misma base de datos se correrá menos el riesgo de que varios procesos utilicen valores diferentes para una misma información.

Otra característica importante de las bases de datos es que su estructura debe ser independiente del formato del dato almacenado. Gracias a esto, en efecto, es posible adaptar la estructura a diferentes exigencias sin tener que modificar todo el sistema.

Para facilitar el uso de una base de datos se proporciona también un lenguaje específico, por medio del cual puede manipularse la base de datos con comandos directos o en forma programada, pero sin tener que recurrir, como antes, a programas escritos en BASIC o COBOL. Este lenguaje es a menudo potente y cómodo para el usuario. Si, por ejemplo, el usuario de un sistema bancario quiere saber cuántos clientes han ingresado cantidades superiores a 500.000 pesetas en un mes en los últimos ocho meses,

no tiene que hacer ya una solicitud por cada atributo, sino que tendrá la posibilidad de especificar todas sus peticiones con un solo comando.

Esto nos lleva a uno de los aspectos más importantes de una base de datos: los diferentes modelos del sistema según el tipo de llaves de acceso deseadas.

Imaginemos, por ejemplo, que el encargado de una ferretería (informatizada, claro) está interesado en el artículo "destornillador de estrella". Es posible que desee saber cuántos destornilladores de estrella quedan en el almacén o cuántos de entre todos los que hay pertenecen a una firma determinada. También podría querer saber qué cantidad de empresas tienen destornilladores de estrella en sus catálogos, o bien, directamente, si la casa X tiene un destornillador de estrella determinado. Como puede verse, incluso en un caso en apariencia tan sencillo hay diferentes solicitudes y, por tanto, distintos métodos de acceso.

Hoy en día los modelos estudiados y utilizados son tres: jerárquico, de red, relacional.

El modelo jerárquico (Fig. 8) presenta una estructura en árbol donde las relaciones entre los términos están reguladas por una jerarquía padre-hijo: un padre puede tener varios hijos, pero un hijo no puede tener varios padres. Este modelo es relativamente fácil de implementar y de utilizar: las relaciones entre los datos

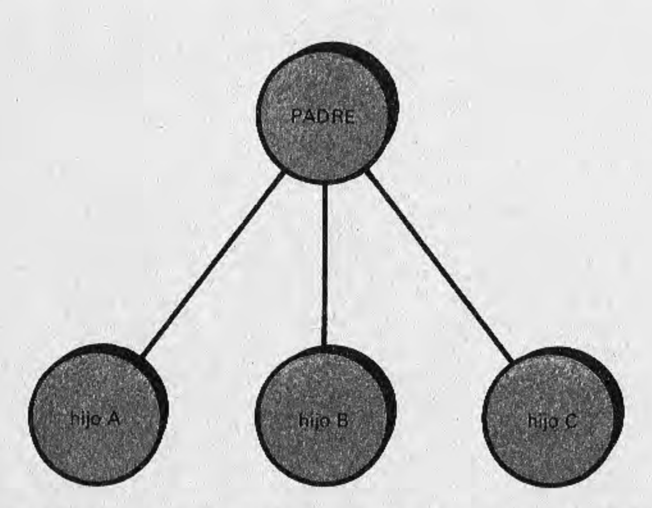


Figura 8.—Base de datos de tipo jerárquico: ordenada de forma estructurada y sencilla, no es aún suficiente para expresar relaciones complejas de muchos-a-muchos.

son claras y bastante sencillas de seguir. Sin embargo, es demasiado rígido y por ello está empezando a caer en desuso.

Para el caso en que haya que expresar relaciones en las que cada hijo posee varios padres (se habla también de relación muchos-a-muchos, en contraposición a la de uno-a-muchos del modelo jerárquico), nació el modelo de red (Fig. 9). Es mucho más flexible que el precedente, pero tiene también su lado negativo debido a que, cuando el número de conexiones crece, la red se hace tan compleja que resulta muy difícil descifrarla.

Se ha desarrollado así un modelo que supera estos inconvenientes y que se asemeja más al modo en que nosotros representamos las estructuras de datos: el modelo relacional (Fig. 10). Este sistema está compuesto por un conjunto de tablas (líneas y columnas) que expresan las relaciones entre entidades diferentes; en la figura mencionada aparece una relación del tipo "herramientas" en la base de datos de una ferretería.

Su función es clara: a cada tipo de herramienta del almacén se la asigna un número de orden, una firma productora, un material de construcción, un precio y la cantidad disponible. Sólo el primer campo de cada línea es único (de forma que sea fácil individualizarlas), todos los demás pueden ser iguales.

El número de columnas indica el grado de la relación. Al cambiar pequeños subconjuntos de las distintas relaciones en juego

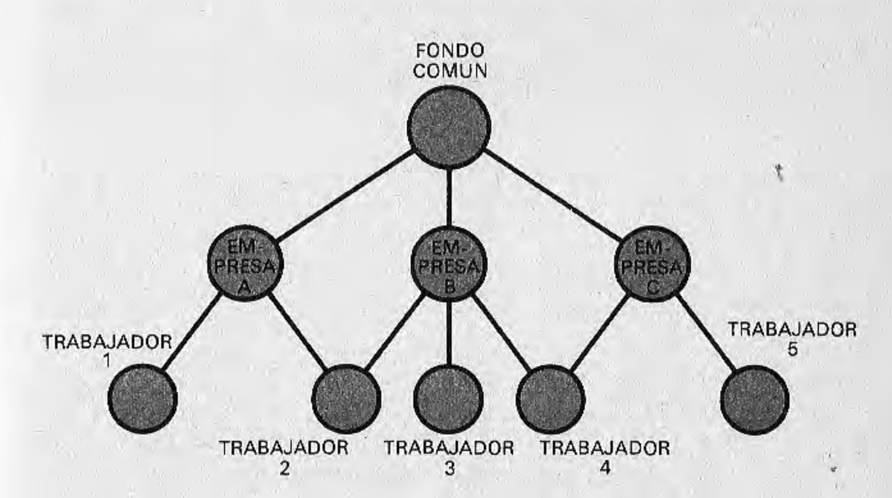


Figura 9.—Base de datos con organización de red: el sistema reticular hace posible tener en cuenta todas las relaciones, pero es demasiado confuso e intrincado y, en casos complejos, resulta casi imposible de descifrar.

Relación: HERRAMIENTAS

NUMERO	EMPRESA	MATERIAL	PRECIO	CANTIDAD	POBLACION
0153	BETA	ACERO	1000	10	LONDRES
10679	BETA	VANADIO	2000	10	LONDRES
80753	USAG	HIERRO	500	5	ROMA
43799	USAG	ACERO	1200	15	ROMA
00152	BETA	CROMO	3000	12	LONDRES
35421	KRUPP	ACERO	1500	5	VALENCIA

Figura 10.—Base de datos de tipo relacional. Se asemeja más fácilmente a los esquemas mentales del usuario y permite, al tiempo, una gestión más veloz de los accesos.

en una base de datos relacional obtenemos varias proyecciones posibles, como la de la figura 11, que facilita la nueva relación Empresa-Dirección.

Este modelo es comprensible con rapidez por parte del usuario y fácilmente modificable, permitiendo todo tipo de combinaciones. Además, los tiempos de búsqueda son mucho menores, al no necesitar ninguna modificación de complicados punteros.

Está empezando a pensarse, incluso a nivel de ordenadores personales, implementar algún tipo de Bases de Datos en el propio sistema operativo, con el fin de permitir al usuario gestionar automáticamente sus Bases de Datos sin tener que recurrir a com-

Relación: EMPRESA-POBLACION

EMPRESA	POBLACION
BETA	LONDRES
USAG	ROMA
KRUPP	VALENCIA

Figura 11.—Ejemplo de proyección de las relaciones. La potencia del modelo relacional estriba en que permite extraer de una (o varias) relaciones de la base de datos visiones diferentes, según las claves de acceso y los más variados criterios.

plejos programas adicionales. Por otra parte, hay en el mercado numerosos y variados paquetes de software para la gestión de bases de datos personales, organizadas incluso según el modelo relacional (dBase II y III).

La gestión de ficheros en el CP/M

Tras haber visto los principios relacionados con la gestión de los ficheros en un sistema de memoria de masa, trataremos ahora de cómo son ejecutadas estas funciones por uno de los sistemas operativos más difundidos hoy en día entre los ordenadores personales: el CP/M. La gestión de ficheros en el CP/M es efectuada por los programas CCP y BDOS.

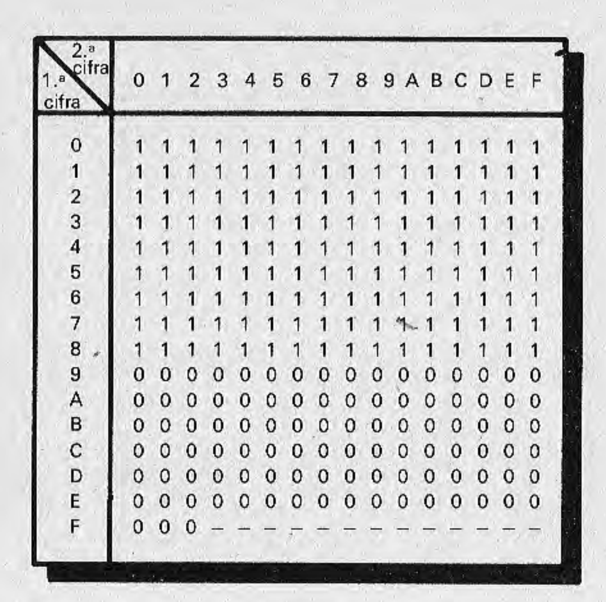
Al haber sido diseñado este sistema operativo con el fin de que se pudiera adaptar a ordenadores con las características más dispares, con la sola condición que la CPU sea un Z80, el tipo de disquete utilizado varía desde el de 8 pulgadas hasta el de 5 1/4, desde densidad sencilla a doble, desde hard hasta soft sectored.

Para que resulte más sencillo, aquí nos ocuparemos de la versión 1.4 del CP/M que utiliza un floppy disk del tipo IBM 3740, de 8 pulgadas, densidad sencilla, soft sectored. El disco está formado por 77 pistas, y cada pista por 26 sectores. El formato de los sectores es de 128 bytes y la correspondencia entre sectores y registros es de 1:1. La operación de formateo organiza el disco de la siguiente manera: las dos primeras pistas contienen informaciones del sistema y un cargador de bootstrap; en la tercera pista, 16 sectores están reservados para el índice de los ficheros, mientras que las restantes pistas están disponibles para el usuario. La máxima dimensión permitida de un fichero es de 256 Kbytes.

El CP/M utiliza para encadenar los sectores pertenecientes a un mismo fichero una lista de sectores, llamada File Control Block. Cada una de estas listas, de una longitud de 32 bytes, direcciona 16 Kbytes. Están normalmente contenidas en disco, en el interior del directorio, pero cuando se utiliza el fichero son cargadas en memoria para permitir así un acceso más veloz.

Cuando hablamos de la organización de los sectores en el disco vimos que no era conveniente disponer los sectores de modo contiguo, ya que había que esperar a que se completara una revolución entera del disco entre uno y otro acceso. Para evitar esto el CP/M utiliza una separación física (gap) de 6 sectores entre un sector "lógico" y otro.

El sistema utilizado para conocer el estado de los sectores es el BIT MAP. Consiste en un grupo de 243 bits, en que cada bit indica el estado de un grupo de 8 sectores contiguos (lógicamente). Mientras al menos uno de los sectores del grupo esté libre, el



1.ª cifra \ 2.ª cifra	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
8	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-



Figura 12.—Bit Map del CP/M, una manera bastante sencilla de conocer el estado de los sectores.

bit correspondiente se mantiene a 0. Cuando todos han sido asignados, el bit que corresponde se pone a 1. En la figura 12 aparece un posible Bit Map del CP/M: hasta el bit 90 todos los sectores están ocupados, y a partir de él, todos están libres.

Veamos ahora cómo funciona el acceso del usuario a un fichero. Cuando el usuario proporciona el nombre del fichero que desa tratar, el BDOS busca este nombre en el File Control Block, (FCB) del disco; si existe, hallará la información relativa a su colocación en el disco (pista-sector) y la transferirá a la memoria, a un segundo FCB creado por el CCP.

Cada vez que se escribe algo en el fichero (para lo que es necesaria una modificación del disco), el contenido del FCB se actualiza. Al concluir la sesión el FCB presente en emoria será copiado sobre el que se halla en disco, actualizando así todas las modificaciones efectuadas.

CAPITULO IV

SOFTWARE DE BASE: MAS ALLA DEL SISTEMA OPERATIVO

Soportes del sistema: con ellos todo resulta más fácil

Como ya vimos en el primer capítulo, el usuario de un ordenador no debe preocuparse hoy en día, como los de antaño, de accionar una serie de interruptores para programar su ordenador o proporcionarle un paquete de tarjetas mediante un lector para ejecutar un programa. Con el modo de funcionamiento interactivo, el usuario, por medio de un terminal de vídeo, puede escribir un programa, corregir los errores, ejecutarlo, solicitar al ordenador más información e incluso utilizar programas incluidos previamente en el sistema operativo para resolver ciertos problemas.

En la actualidad el usuario no tiene por qué saber cómo gestiona el ordenador la memoria o los discos, o cómo recibe los datos; lo único que pretende es utilizar el sistema de la manera más sencilla posible para obtener resultados. Es decir, el ordenador debe ser un medio y no un fin: si un usuario tiene que resolver un problema, no debe tener que pasar todo el tiempo disponiendo el ordenador para que funcione según sus necesidades, sino que debe poder utilizarlo inmediatamente y sin demasiadas dificultades. Esta es la finalidad del software de base (o software del sistema), *que comprende el sistema operativo* y otros muchos programas de soporte.

Para este propósito, un software de base moderno debe estar estructurado de manera que ofrezca a los usuarios al menos dos tipos de facilidades:

— soportes para el programa en ejecución,
— soportes para el usuario.

Los primeros se encargarán de proporcionar al usuario todos los medios que necesite durante la ejecución: memoria, accesos a discos, entrada y salida de datos, impresión, etc. Mientras los segundos deben facilitar la realización de cualquier otro deseo del usuario. El programa deberá ser cargado en memoria sin que el usuario tenga por qué buscar un área libre y colocarlo dentro. Cuando un dato tenga que ser insertado en un disco será el sistema operativo el que se preocupe de organizar todas las operaciones: búsqueda del nombre del programa en el índice del disco, direccionamiento de la pista-sector correspondiente (véase capítulo 3), localización en memoria de un área libre para la transferencia, ocupación del área y posterior transferencia.

Vemos que, incluso en este caso, hay toda una larga serie de operaciones que harían muy compleja la labor del programador si no existieran los programas de servicio, soporte o asistencia.

Otro de los soportes esenciales que el software de base debe proporcionar es el control de la seguridad. Podría suceder que el usuario, intencionadamente o no, quisiera acceder a programas o áreas prohibidas, lo que produciría problemas o daños más o menos serios. Para evitar esto, el software de base se ocupa de controlar toda operación efectuada por los usuarios o por sus programas.

En relación con los soportes proporcionados al usuario, se está imponiendo últimamente la filosofía de que sean "amigables", es decir: serviciales, fáciles de entender y prácticos en las fases de creación, verificación y funcionamiento del programa. Como estas asistencias dependen de la empresa que produce o vende el ordenador, no es posible establecer un producto estándar. Veremos a continuación los de uso más corriente.

Los Editores

Para componer un programa o un texto, el usuario debe tener a su disposición un instrumento capaz de aceptar los caracteres escritos y memorizarlos a continuación en disco.

En los albores de la programación (es decir, hasta hace pocos años...) el programador tenía que teclear en una máquina perforadora un paquete de tarjetas con todas las instrucciones e introducirlo posteriormente en el dispositivo lector. Al cabo de un tiempo, que podía ser incluso el día siguiente, el ordenador señalaba los errores cometidos y, con toda probabilidad, el programador tenía que repetir todo de nuevo. Este método, además de ser

terriblemente pesado, reducía la posibilidad de manejar los ordenadores a unos pocos expertos privilegiados de bata blanca.

Hoy en día, por suerte, todos los ordenadores poseen un programa que permite al usuario escribir un texto, revisarlo durante la composición, corregir los errores y, por último, almacenarlo. Estos programas, denominados "Editores", se hallan disponibles en varias versiones y están en constante evolución para que un número cada vez mayor de usuarios puedan utilizarlos.

Veamos ahora las características de estos programas, nacidos como auxilio al usuario para permitirle escribir textos con más facilidad y flexibilidad, y que están ahora cada vez más orientados hacia la edición de textos (Text Editor), es decir, hacia la composición de textos destinados a su impresión.

Los Editores se dividen en dos grandes categorías: orientados hacia la línea y orientados hacia el carácter. De estos últimos derivaron posteriormente los paquetes comerciales conocidos por el nombre de Word Processor (literalmente procesador de palabras, aunque se les suele conocer como procesador o tratamiento de textos, aun con la ambigüedad que esto supone). Para entender esta distinción, vamos a explicar cómo funcionan en la práctica estos programas.

Un Editor, una vez reclamado por el usuario, por regla general con el comando EDIT, se ocupa de reservar una zona de me- destinada a acoger los caracteres escritos, pregunta cómo deseamos que se llame el programa, reserva en el disco un fichero para este fin, deposita los caracteres en la zona de memoria reservada expresamente y, una vez concluida la sesión de edición, se encarga de almacenarlo todo en disco.

En realidad, además de éstas, que son las principales funciones, el programa debe desarrollar, en estrecha relación con el sistema operativo, toda una serie de operaciones diferentes. Las principales son:

— una vez iniciada la edición se ocupa de comprobar si el programa que el usuario quiere "editar" es nuevo o existe ya; en el segundo caso deberá encargarse de hacer una copia de seguridad antes de permitir que el usuario escriba encima;
— a la hora de acceder a un programa compartido se crean problemas de seguridad: no todos los usuarios pueden, por ejemplo, escribir en los programas ajenos. Existen reglas de protección que disciplinan los accesos, y el Editor, por medio del sistema operativo, no permite el acceso a los trabajos protegidos;
— en un sistema multiusuario podría suceder que dos usuarios quisieran acceder simultáneamente a un mismo fiche-

ro. También en este caso el Editor impedirá el segundo acceso: sería una buena "broma" que mientras alguien estuviera escribiendo un programa, otra persona le fuera cambiando las instrucciones...

La distinción que hemos señalado anteriormente acerca de editores orientados hacia la línea o hacia el carácter radica en la forma de presentar un texto y, por tanto, en el modo de operar del usuario.

Los Editores orientados a la línea permiten escribir, borrar y corregir caracteres en el interior de la línea en cuestión. Tras haber pulsado la tecla de "enter" o "carriage return", el renglón se hace definitivo, aunque sigue apareciendo en pantalla.

Para hacer una corrección es necesario disponer el modo de funcionamiento "comandos", situarse en la línea que se desea corregir especificando su número, expresar con un comando los caracteres que se desea sustituir y efectuar el cambio. Como puede verse, se trata de un procedimiento muy rebuscado y que hace perder mucho tiempo a la hora de examinar el texto para hallar y corregir los errores. Este tipo de Editores resulta particularmente abandonado, sobre todo cuando hay que componer textos, introducir nuevos fragmentos, hacer correcciones o desarrollar las diversas operaciones que son normales en tipografía.

Han nacido así los Editores orientados al carácter, llamados también "screen editor" (editor de pantalla), que permiten, por medio de cuatro flechas (arriba, abajo, derecha, izquierda) o de un "paddle" (palanca manipuladora), mover a placer el cursor a lo largo del texto y efectuar todas las operaciones necesarias. Está claro que resulta muy fácil para el usuario, sobre todo si dispone de un "paddle", seguir el texto escrito en la pantalla y hacer correcciones: es como leer la página de un periódico sirviéndose del dedo conforme se va leyendo.

Uno de los métodos para componer textos actualmente en el mercado es el del nuevo super personal de Apple: el Macintosh. Su Editor es también gráfico: permite poner en medio del texto ciertas figuras que el operador ha dibujado en la pantalla por medio del "paddle" (que en este caso es el conocido ratón "mouse"). Además, tratando siempre de hacer el ordenador lo más funcional posible, los comandos disponibles están visualizados gráficamente en los márgenes de la pantalla: de esta manera, hasta un usuario que desconozca la terminología utilizada es capaz de ejecutar las operaciones deseadas.

Tenemos que hacer una distinción entre los "pseudo-editores" BASIC, disponibles en los primeros ordenadores personales y micros domésticos más económicos, y los verdaderos editores, que se encuentran junto con los sistemas operativos dignos de tal

nombre y ya implementados en casi todos los ordenadores personales (como el CP/M y similares). Vamos a examinar el IBM-PC que posee ambas posibilidades: un modo de funcionamiento BASIC, en el que pueden escribirse programas en lenguaje BASIC y utilizar sus funciones para gestionar el ordenador, y el editor EDLIN, que permite escribir programas en cualquier lenguaje.

El editor BASIC está prácticamente funcionando siempre y considera "programa" todo aquello que ve en la memoria. Para iniciar un nuevo trabajo es necesario utilizar el comando NEW, que destruye el antiguo programa presente en memoria y prepara el Editor para aceptar otros nuevos. Este Editor está orientado hacia la línea, por lo que memoriza una línea cada vez. Hay que poner un número al comienzo de cada línea, pues de otro modo el BASIC interpreta la instrucción que se ha escrito como un comando que hay que ejecutar de forma directa y lo lleva a cabo nada mas pulsar la tecla <enter>. En modo indirecto, en cambio, el BASIC acepta y memoriza todas las instrucciones, precedidas por el número de orden escrito. En el interior de cada línea podemos efectuar todas las modificaciones que deseemos por medio de las flechas de desplazamiento horizontal.

Si queremos insertar una nueva palabra en medio de las que ya existen podemos utilizar el modo INSERT. Con el comando INS el ordenador permite introducir un carácter nuevo, desplazando los otros hacia la derecha. Si lo que queremos es modificar líneas de programa ya existentes, tendremos primero que visualizarlos en la pantalla y operar luego los cambios.

Para visualizar un programa o parte de él hay dos posibilidades, materializadas en los comandos LIST o EDIT. Por medio de estos comandos podemos visualizar una o varias líneas. Utilizando las flechas de control del cursor nos posicionamos sobre los caracteres que queremos cambiar y escribimos encima los correctos (o añadimos otros con la tecla INS). Pulsando <enter> la línea será modificada también en la memoria. Para borrar una línea bastará activar el comando DELETE, especificando el número de la línea o líneas que se quieren cambiar.

Una vez terminada la edición, el programa se podrá guardar en cinta o disco por medio del comando SAVE. De esta manera será posible conservarlo de forma permanente y volver a utilizarlo reclamándolo en memoria con el comando LOAD. Una vez que el programa se halle en memoria se procederá a su ejecución con el comando RUN.

Este Editor de BASIC es, como vemos, muy sencillo y esencial, y sólo permite la edición de programas en BASIC. Para otras exigencias tenemos a nuestra disposición el Editor EDLIN.

Este, en cuanto a su uso, es bastante parecido al anterior, pero permite editar textos con formatos más generales. También en

este caso cada línea está caracterizada por un número, pero ahora no forma parte del programa: la numeración es realizada por el Editor y sólo sirve para hacer referencia a las líneas sin ambigüedades. El número de línea está comprendido entre 1 y 65529.

El EDLIN reside en disco y opera sobre programas contenidos en disco. Si el fichero que vamos a editar está ya creado lo carga en memoria desde el disco y, al acabar, señala el fin del fichero de entrada; si no existía nos lo indica con el mensaje "New File" (nuevo fichero).

El Editor se presenta en modo "comandos" al usuario con un "*". Para escribir el texto hay que pulsar una I: se sitúa así en INSERT mode y todo lo que le introduzcamos será memorizado. El Editor, por su parte, escribe en la pantalla el número correspondiente a la línea que el usuario está editando: como ya dijimos, este número no tiene ningún significado, en realidad, en el contexto del programa.

Con el comando CTRL BREAK termina el INSERT mode y EDLIN vuelve a presentar el carácter "*", indicando el funcionamiento en modo comandos. En esta situación el usuario puede ejecutar una serie de manipulaciones en el texto:

- búsqueda de una secuencia (SEARCH): el Editor busca una secuencia de caracteres determinada proporcionada por el usuario a lo largo del texto, parándose en la primera línea en donde la encuentra. Esto resulta muy útil cuando, al volver a tratar un programa editado anteriormente, se quiere acceder a cierto punto sin tener que volver a leerlo todo;
- visualización del texto (LIST): permite leer en la pantalla las líneas del programa compuestas anteriormente sin modificarlas;
- sustitución de un grupo de caracteres (REPLACE): permite la sustitución de un grupo de caracteres por otros, operando sobre todo el texto. Si, por ejemplo, tenemos que cambiar en todo un programa el número 1000 por el 999, con este comando se ahorra la búsqueda y corrección todas las veces que aparezca el número 1000;
- comprobación de una línea: pulsando el número de una línea, el Editor presentará su contenido en la pantalla permitiendo al usuario su modificación;
- fin de la edición sin actualizar en disco los cambios realizados (QUIT);
- fin de la edición (END): con este comando, el Editor salva en disco el contenido de la memoria.

Este Editor es, por tanto, mucho más potente y versátil que el precedente.

Hay además otros Editores que pertenecen a las categorías de los "Screen Editor", más conocidos con el nombre de WORD PROCESSOR (procesadores de textos, llamados también sistemas de videoescritura). Estos permiten componer textos para fines tipográficos y proporcionan en la pantalla la imagen de la página, con la posibilidad de efectuar correcciones e inserciones, desplazándose hacia arriba o hacia abajo.

En los ordenadores personales mejores hay ya productos de este tipo, más elásticos y potentes aún que los Editores utilizados en los grandes ordenadores. Citamos entre éstos el Word Star, programa que permite componer tanto textos como programas.

Traductores e intérpretes

Ahora que conocemos los medios utilizados para escribir los programas, veamos qué nos ofrece el ordenador para verificarlos y ejecutarlos.

Un ordenador funciona por medio de una unidad central que ejecuta una serie de operaciones definidas por secuencias de "0" y "1" (códigos binarios). El escribir programas expresados en forma binaria (o hexadecimal) es una operación de lo más compleja y larga, de modo que se ha tenido que recurrir a algún medio para simplificar el trabajo de los programadores.

Así nacieron los lenguajes ensambladores (Assembly o Assembler) y los programas ensambladores, que son los que transforman los símbolos nemotécnicos (o nemónicos) que constituyen las instrucciones en lenguaje ensamblador a código máquina.

Los ensambladores poseen cuatro ventajas de interés en relación con los códigos binarios o hexadecimales del lenguaje máquina:

- están compuestos por símbolos nemotécnicos y no por códigos numéricos del lenguaje máquina;
- las direcciones son simbólicas y no absolutas;
- son mucho más fáciles de leer;
- resulta más fácil incluir datos en el programa.

Estos lenguajes imitan bastante fielmente la estructura del ordenador y, gracias al uso de símbolos nemotécnicos para las instrucciones, permiten escribir y leer con mayor facilidad los programas. El programa ensamblador se encargará más tarde de traducir a lenguaje máquina las instrucciones escritas por el programador, obteniendo unos códigos comprensibles al instante para la unidad central.

La otra gran ventaja de no utilizar el lenguaje máquina reside en que el programador no tendrá que preocuparse de las direcciones a las que hace referencia en la confección del programa, pues será maś tarde el ensamblador (y a continuación el "Linker") quien ordenará y calculará convenientemente estas direcciones.

El nacimiento de los ensambladores ha traído asociado el de los desensambladores. Estos programas, como su propio nombre sugiere, cumplen la función inversa, es decir, pasan el código ejecutable de un ordenador al lenguaje ensamblador. Resulta fácil adivinar la utilización que se les puede dar a estos programas: si los aplicamos al código de un programa ejecutable (por ejemplo, un juego o una parte de sistema operativo), de por sí indescifrable al ser una larga sucesión de números, nos hallaremos frente al mismo programa, pero en forma simbólica, resultando así mucho más comprensible. De esta manera, un buen programador será capaz de entender cómo funciona el programa e incluso, si lo desea, de adaptarlo a sus necesidades. Esto explica porqué los programas desensambladores no son siempre entregados junto con el sistema operativo... ¡sería la anarquía total!

```
        MULT; Multiplicación entre dos
              enteros de 16 bit
              ; sin signo
              ; Multiplicación en HL
              ; Multiplicando en DE
              ; Resultado en HL
        LD B, 16
        LD C,D
        LD A, E
        EX DE,HL
        LD HL,0
MLOOP:  SRL C
        RR A
        JR NC,NOADD-$
        ADD HL,DE
NOAD:   EX DE,HL
        ADD HL,HL
        EX DE,HL
        DJNZ MLLOP-$
        RET
        END
```

Figura 1.—Subrutina de multiplicación para la CPU Z80.

De todos modos, en muchas ocasiones también los lenguajes ensambladores resultan insuficientes para las exigencias de programación. Baste como ejemplo observar la figura 1: se muestra la manera en que es implementada una simple operación de multiplicación, escrita en el lenguaje Ensamblador del Z80, es decir, asociado a la CPU Z80. Como se ve, resulta bastante incomprensible y, a menos que se trate de un experto, no resulta fácil descubrir que esa larga sucesión de siglas corresponde a una sencilla multiplicación entre dos números. Por este motivo han nacido los lenguajes de alto nivel, que expresan un modo de programar el ordenador más comprensible para el hombre y más sencillo por tanto.

Los lenguajes de alto nivel presentan las siguientes ventajas:

— se requiere menos tiempo para aprenderlos. Esto determina que cada día los nuevos programadores se aclimaten más rápidamente al ambiente de trabajo y que empiecen a producir más velozmente. Por otra parte, los usuarios de ordenadores personales aprenden con rapidez a hacerse sus propios programas;
— más facilidad para eliminar errores. En efecto, no es necesario recurrir a los extraños trucos que el lenguaje ensamblador requiere, ni es tampoco indispensable conocer la estructura del ordenador que se está utilizando;
— mayor potencia expresiva, que hace posible un lenguaje orientado hacia el problema y no hacia la máquina. Así, secuencias de operaciones como las de la figura 1 son recopiladas en una sola instrucción, lo que da lugar simultáneamente a que disminuyan las probabilidades de error;
— documentación más sencilla. Un programa escrito en assembler requiere, por regla general, la presencia de un manual que explique su funcionamiento; en cambio, si el programa está bien escrito en un lenguaje de alto nivel, permite que se sobreentiendan muchos de sus mecanismos de funcionamiento;
— relativa independencia del ordenador. Hoy en día muchos de los programas de alto nivel funcionan en todos los ordenadores disponibles. Así, si escribo un programa en Pascal para el IBM-PC tendré la posibilidad, con pocas variaciones formales, de adaptarlo para el Apple, que utiliza una CPU totalmente diferente. Este factor es una gran ventaja para la economía de los programas. No podré, en cambio, hacer lo mismo con un programa escrito en ensamblador, ya que este lenguaje reproduce fielmente las características de la CPU para la que ha sido escrito;
— mayor velocidad a la hora de confeccionar un programa.

Como el número de instrucciones que hay que escribir es muy inferior y hay más estructuración y linealidad en el programa, resulta obvio que el tiempo empleado se reduce considerablemente.

Los programas escritos en un lenguaje de alto nivel han de ser tratados por otros, llamados compiladores, que se encargan de traducirlos al código que la CPU de ese ordenador en concreto utiliza. El principal fallo que presentan estos compiladores que, al tener que resolver situaciones de carácter general, el código producido por ellos no está optimizado. Por tanto, un buen programador en ensamblador es capaz de resolver un mismo problema con un programa más veloz y que ocupa menos espacio en la memoria. Esto, sin duda, es cierto: un compilador produce un código ejecutable que ocupa, de media, entre dos y tres veces el espacio estrictamente necesario. Pero hay que tomar en consideración también dos factores importantes:

- cuando un programa es muy vasto y complejo resulta muy difícil que incluso un buen programador en ensamblador logre dominar la situación y optimice el programa;
- para programas muy amplios el coste, tanto de personal como de tiempo empleado, resulta tan alto en la versión ensamblador que es preferible la desventaja ofrecida por el compilador.

Para programas verdaderamente complicados, como puede ser el caso de un sistema operativo, hoy en día se prefiere escribir todo en un lenguaje de alto nivel para poder probarlo en un tiempo razonable. La experiencia indica que tan sólo el 20% del programa es crítico bajo el punto de vista de la velocidad de ejecución. Una vez localizados estos puntos críticos lo que se hace es reescribir sólo estas partes en assembler, dejando el resto escrito en lenguaje de alto nivel.

Los compiladores disponibles en la actualidad para los ordenadores personales son capaces de traducir casi todos los lenguajes utilizados en los grandes ordenadores. Esto se ha conseguido gracias a la enorme difusión del sistema operativo CP/M primero y del MS-DOS después, que poseen casi todos los lenguajes más extendidos. Los más conocidos son:

- FORTRAN: es el lenguaje de alto nivel de gran utilización más antiguo;
- COBOL: el más utilizado en el campo de la gestión (caracterizado por un exceso de palabrería tal que los programas escritos en este lenguaje baten todos los récords en longitud...);

- PASCAL: el primer lenguaje inspirado en los principios de la programación estructurada moderna es, con toda seguridad, uno de los más comprensibles y autodocumentados entre los existentes hoy en día.
- C: verdadero lenguaje para la programación estructurada. Es "amistoso", flexible, muy eficiente, potente y portátil (intercambiable). Permite acercarnos con sencillez, cuando lo necesitemos, al ensamblador. En él está escrito gran parte del Unix, así como compiladores e intérpretes de muchos otros lenguajes (incluso la animación de las escenas de "El retorno del Jedi" se programó en C).

Hay además otros lenguajes que empiezan a difundirse entre los ordenadores personales. Uno de los ejemplos más interesantes es el LISP. Está disponible para el Apple y en sistema operativo CP/M y es el primer escalón hacia las aplicaciones de Inteligencia Artificial.

Tratamiento aparte merece el BASIC, que es, con toda seguridad, entre los lenguajes de alto nivel, el más fácil de aprender: no es casualidad que esté implementado en todos los ordenadores personales.

Pero posee también sus grandes desventajas; la falta de estructuración lógica que lo caracteriza lo hace incómodo bajo el punto de vista de tratamiento de los datos y de la comprensión de los programas. En efecto, en la creación de programas muy complejos la utilización frecuente de saltos incondicionales (GOTO) y la imposibilidad de usar una estructura adecuada hace a menudo difícil descifrarlos. Para programas sencillos e inmediatos sigue siendo, sin duda alguna, el lenguaje más eficaz.

Una característica típica del BASIC (utilizada a veces por otros lenguajes de alto nivel) es la de no ser compilado sino interpretado. Un intérprete es un programa que, residiendo en memoria, traduce directamente las instrucciones del lenguaje a código máquina según va ejecutando el programa.

El intérprete, al tener que efectuar cada vez que ejecuta una instrucción su traducción (mientras que los compiladores lo hacen de una vez para siempre) es, por regla general, más lento que un compilador, pero presenta indudables ventajas. En primer lugar, es de uso muy inmediato: el usuario ni se da cuenta de la traducción ni está ocupado en las a veces complejas operaciones de compilación u otras tareas que éste requiere a menudo (linker, loader, etc.). Además, al no producir código ejecutable del programa, no necesita amplias áreas de memoria para almacenarlo. Por este motivo se prefieren los lenguajes interpretados, como el BASIC, para implementaciones en ordenadores personales que,

en sus versiones más económicas, no tienen precisamente una abundante memoria.

Al estar muchas veces memorizado sobre una EPROM, además, el BASIC interpretado no necesita discos ni cintas, sino sólo memoria suficiente para almacenar el programa editado por el usuario. Como no se produce código máquina, toda la memoria está destinada únicamente a contener el texto tecleado por el usuario y los valores de las variables del programa.

Cargadores (loaders) e instrumentos de comprobación

Como hemos visto anteriormente, los programas escritos por el usuario son traducidos a lenguaje máquina por ensambladores y compiladores. En cambio, el "cargador" (loader) es un programa complejo que toma textos en código de máquina (o, con más propiedad, código objeto) y los prepara para ser ejecutados en el ordenador.

Las funciones del cargador son fundamentalmente cuatro:

— reservar espacio de memoria para el programa (ubicación);
— conectar entre sí todos los módulos de programas diferentes (linking);
— volver a calcular todas las direcciones no absolutas, es decir, aquellas que dependen de la ubicación del programa (reubicación);
— cargar el código de ejecución y los datos en memoria.

En la actualidad, estas funciones, tanto en los grandes ordenadores como en los personales, son hechas en dos fases: la fase de enlace (link) y la de ejecución.

Vamos a examinar, a modo de ejemplo, de qué manera funciona en este aspecto el CP/M, que opera de un modo muy parecido al de otros muchos sistemas operativos.

Cuando el operador escribe programas en ensamblador o en lenguajes de alto nivel no se preocupa en asignar direcciones absolutas, sino que se limita a poner unas etiquetas (label) a las que se referirá en los saltos, en las llamadas a subrutinas o en el empleo de las tablas de datos, es decir, utiliza unas referencias "relativas". Además, es corriente dividir el programa en varios módulos o utilizar partes de programa de librería (comentaremos esto en un próximo apartado), por lo cual será necesario después hacer las oportunas conexiones entre las distintas partes del programa.

El que se ocupa de efectuar estas conexiones es el programa enlazador (linker). Al especificar en el comando LINK todos los nombres de los programas, ya compilados o ensamblados, que se quieran utilizar, éstos son unidos y se produce un código, ejecutable directamente por la máquina. Es el mismo código objeto del que hemos hablado anteriormente, pero en él han sido ya efectuadas las conexiones entre los distintos programas, y las etiquetas que compiladores y ensambladores habían dejado, digámoslo así, en el aire son "materializadas".

El linker se ocupa, por tanto, de organizar las direcciones del programa para la ejecución. Normalmente, el sistema operativo CP/M carga el programa en la dirección 103 hexadecimal, pero también es posible especificar una dirección diferente para el programa en sí (código ejecutable) y para los datos.

Por medio de las opciones /P: <dirección> y /D: <datos> el CP/M permite al usuario que fije la dirección a partir de la cual quiere que el programa sea cargado. Pero tendrá que prestar mucha atención al especificar estas direcciones para no eliminar cosas que ya existen (y aún sirven) o cargar el programa en una zona de memoria que no existe... En efecto, el linker no ejerce controles de protección y sigue con el proceso a pesar de que se hayan cometido errores de este tipo.

Otra opción útil ofrecida por el CP/M es la lista de todas las referencias globales de los programas: al especificar /M el linker dará las direcciones de todos los símbolos que hemos declarado globales (como tablas de datos, puntos iniciales de subrutinas y otros parecidos). Este factor resulta muy útil en la fase de verificación del funcionamiento de un programa, que trataremos más adelante.

Una vez obtenido el código ejecutable se da el comando de ejecución (/G en el CP/M) y el sistema operativo se encargará de cargar el programa en memoria y de ejecutarlo.

Quienes hayan intentado ya escribir un programa sabrán que en la primera ejecución, y casi con total seguridad..., no funcionará. La manera de subsanar este pesado e inevitable inconveniente es la de repasar con toda paciencia el programa, tratando de descubrir qué es lo que falla.

Si los programas son muy largos y complejos, esta labor es particularmente difícil. Algunos sistemas operativos proporcionan un programa llamado debugger (corrector, depurador). Gracias a él es posible seguir el programa paso a paso, ejecutando una instrucción cada vez, lo que facilita la localización del punto de funcionamiento anómalo.

En los primeros sistemas operativos y ordenadores personales más sencillos, el único medio de que se disponía era examinar el contenido de la memoria; el usuario podía bloquear la eje-

cución del programa (por medio, por ejemplo, de una tecla de interrupción) y, a continuación, visualizaba el contenido de la memoria del ordenador (hacia un "dump") para tratar de descubrir lo ocurrido. Con esto se comprende la gran utilidad de la opción vista del linker, que permite saber las direcciones de los símbolos del programa. En efecto, si utilizamos unas variables (es decir, celdillas de memoria cuyo contenido varía durante la ejecución del programa) viendo sus valores será posible hacerse una idea acerca del funcionamiento del programa. Un ejemplo puede ser una instrucción que escribe un "1" lógico en una celdilla al comienzo de cierta parte del programa, o diferentes valores según el recorrido efectuado en un salto condicional. Gracias a la ejecución paso a paso, el usuario será capaz de comprobar el recorrido efectuado por el programa.

Otro programa de soporte es el "trazador" (tracer); con él se simula la ejecución de las instrucciones, visualizando sobre la marcha el contenido de los registros del procesador. Como en realidad tal simulación es a menudo más lenta que la ejecución del programa, en muchos casos se ofrece una tarjeta adicional que ejerce la función "tracer" con un procesador dedicado. Este método resulta bastante interesante, ya que el usuario, al poder seguir simultáneamente los datos de pantalla y su programa, distinguirá al instante una posible anomalía.

De todos modos, los más modernos sistemas operativos ofrecen unos depuradores o debuggers interactivos, con los que el usuario puede manipular su programa, pararlo y ponerlo en marcha de nuevo, examinar las estructuras de datos en memoria y los registros de la CPU y modificar su contenido. Si se conoce la dirección de la instrucción en la que se supone que hay algún mal funcionamiento, puede introducir (al finalizar algún paso crítico) un "punto de ruptura" (break point), es decir, un punto en el que el programa cesa su ejecución y cede el control al operador, que podrá realizar todas las operaciones necesarias en la memoria y en los registros de la CPU.

Un elemento que resulta muy útil pero que, por el momento, está sólo disponible para grandes ordenadores, es el debugger simbólico, que ejecuta todas las funciones vistas hasta el momento, pero operando directamente sobre los símbolos y no sobre las direcciones. Esto resulta indispensable si se desea comprobar de una forma rápida y eficaz el funcionamiento de los programas escritos en un lenguaje de alto nivel, en el que, naturalmente, todo es simbólico y, en consecuencia, resulta imposible introducir "break points", ya que cada instrucción es, por lo general, traducida mediante varias palabras del lenguaje máquina.

Queremos hacer notar que los lenguaje interpretados consienten una discreta depuración (debugging) con mucha facilidad. Por ejemplo, en BASIC, al ejecutar un programa se obtiene un mensaje típico (SYNTAX ERROR) cada vez que el intérprete tropieza con una línea sintácticamente incorrecta, parándose en esa línea. Así es posible corregirla en el momento y hacer partir de nuevo el programa (con el comando CONT). En relación con los errores de naturaleza lógica resulta buena táctica intercalar, durante la fase de comprobación, líneas de STOP que contengan al menos algunas instrucciones para la impresión de los datos críticos (cuando el programa se pare será posible revisar todo lo que deseemos y continuar de nuevo con el comando CONT). Todo esto sin tener que comprar potentes pero costosos debuggers y sin arriesgarnos a volvernos locos con nuevas compilaciones, aburridas y complejas. Sólo habrá que prestar atención para evitar que alguna línea errónea no sea descubierta, lo cual puede suceder si pertenece a un ciclo que se ejecuta sólo en contadas ocasiones (los compiladores verifican todas las instrucciones en cada intervención).

DDT, el depurador del CP/M

Vamos a ver ahora uno de los depuradores más utilizados en los ordenadores personales: el DDT (Dynamic Debugging Tool), proporcionado por el CP/M (hay otro parecido para el MS-DOS).

Con el comando DDT <nombre del programa> se carga en memoria el programa que hay que depurar y se pone bajo el control del DDT, que se presenta con "-". es posible escoger entre dos modos de funcionamiento: directo y paso a paso (trace).

En el modo directo se especifican uno o más puntos de parada (break point) y, por medio del comando GO, se inicia la ejecución del programa. Este se detendrá cuando encuentre el break point; entonces podremos examinar el estado de los registros y modificar su contenido con el comando "X" (eXamine). También se podrá comprobar el contenido de la memoria con el comando D (Display), que la presenta en líneas de 16 bytes cada vez. Otro comando útil es el MOVE: desplaza un bloque, de la longitud que especifiquemos, de una dirección a otra. En cambio, con el comando FILL podemos escribir una constante de un byte en toda un área de memoria; resulta muy útil, por ejemplo, para poner a cero una tabla.

El modo paso a paso (trace) hace proceder la ejecución del programa instrucción por instrucción, visualizando a cada paso el estado de los registros de la CPU, aunque, naturalmente, también es posible hacer ejecutar cierto número de instrucciones de una manera automática y controlar el estado de los registros después.

Otro comando que resulta muy útil es el "L" (List), ya que permite desensamblar hasta doce líneas de programa. Además, visualiza los nemotécnicos de las instrucciones assembler, permitiendo al programador conocer qué significado tienen esos números que ocupan las celdillas de la memoria que se están examinando.

Si el usuario quiere introducir otro código en el programa puede, por medio del comando "A" (Assembler), incorporar instrucciones, ya codificadas en lenguaje máquina, en la dirección deseada. Posteriormente podrá comprobar la exactitud de la corrección con el comando "L".

Comos hemos visto, este instrumento de comprobación resulta muy útil y versátil. Por desgracia, el DDT también tiene un inconveniente, que es el de haber sido escrito para el microprocesador 8080 Intel y, por tanto, sólo funciona con el código de éste (o con otro compatible). Es obvio que, si se cambia de procesador, hará falta otro debugger, como el citado anteriormente del MS-DOS.

Librerías de programas

Existen unos soportes ofrecidos por el sistema operativo que tratan de evitar que muchos usuarios pierdan tiempo inútilmente. En efecto, muchos de nosotros nos esforzamos en escribir programas que hagan unas determinadas funciones, que son de uso muy general. Para evitar esto hay ciertos paquetes (package) de software que permiten que el usuario utilice sus funciones simplemente con una llamda desde su programa. Su conjunto es lo que se denomina "librería" de programas.

Hay disponibles, por lo general, muchísimas funciones matemáticas para los lenguajes ensambladores que, de por sí, no las poseen: multiplicación, división, raíz cuadrada, elevación a potencia, etc. También son disponibles pequeños programas que permiten utilizar muchas de las funciones de uso interno del sistema operativo: acceso directo a disco, manejo de I/O...

Naturalmente, los lenguajes de alto nivel disponen, por norma general, de las funciones aritméticas elementales, pero a veces, sobre todo para usos científicos, se requieren funciones analíticas que pueden ser muy complejas. Así, hay funciones trigonométricas (seno, coseno, etc.), logarítmicas, para el cálculo matricial, cálculo vectorial, resolución de sistemas de ecuaciones de varias incógnitas, funciones estadísticas de lo más raras, etc.

Una de las funciones más solicitadas para cálculos científicos y financieros es el uso de operaciones en coma flotante. Existen

paquetes de aritmética con cálculos en coma flotante muy eficaces, pero bastante lentos. Por este motivo suele incorporarse un módulo hardware, especializado en el procesamiento directo de estas funciones a menudo por medio de un co-procesador específico que se incorpora a la CPU (éste es el caso del co-procesador Intel 8087, opcional en el PC IBM). Y todo esto sin tener necesidad de programas de librería que simulen su funcionamiento: el ahorro de tiempo es considerable.

Un campo en el que el usuario de ordenadores personales empieza a encontrar las librerías de servicio muy útiles es el de los gráficos. Ya hay disponibles programas que dibujan curvas, rectas, figuras y otras funciones sin que el usuario tenga que hacer ningún tipo de cálculo, sólo debe indicar lo que desea.

Otro sector muy solicitado es el del Word Processing, o tratamiento (procesamiento o composición) de textos. Resulta muy útil poder disponer de un ordenador para componer cartas, artículos o libros. Los textos son compuestos como si de programas corrientes se tratara por medio del teclado; el procesador de textos se encargará a continuación de encuadrar el escrito, organizando los márgenes derecho e izquierdo, superior e inferior, el espacio entre líneas, el número de caracteres por línea... Los más sofisticados llegan a generar incluso el índice de todos los párrafos, el índice analítico de las palabras, la lista de las figuras y tablas, en resumen, la estructura completa de un libro.

Conclusión

Además de lo visto, queda aún por comentar el vasto mundo de los paquetes de aplicaciones, bien para la automatización de oficinas y gestión de personal (desde las hojas electrónicas a las bases de datos personales), para comunicación, así como los tan de moda paquetes integrados (como el célebre Lotyus 1-2-3), etc., todos ellos mucho más difundidos en los ordenadores personales que en los de grandes dimensiones.

El hecho de que nos quedemos en los procesadores de textos se debe, en primer lugar, a motivos de espacio, y, en segundo, que estos temas serán tratados con mayor amplitud en posteriores volúmenes de la colección. Tal vez la elección haya dependido del hecho de que el Word Processor es descendiente de los editores, presentes originalmente en los sistemas operativos más modernos de los ordenadores, tanto grandes como medianos. Hay, por tanto, una continuidad entre la macro y la microinformática. Éste ha sido el factor que nos ha orientado a tratar, en un libro de divulgación, las peculiaridades de varios sistemas operativos, po-

niendo en evidencia sus diferentes características. Algunas de éstas no se hallan presentes todavía en los ordenadores personales, pero hay que tener presente que la próxima generación de ordenadores personales está adoptando CPUs cada vez más potentes y con espacios de memoria mucho más amplios, lo que hace posible la comercialización de muchas de estas sofisticaciones también en microordenadores. Asimismo es importante señalar que:

- algunas de estas funciones no tienen gran utilidad práctica, al menos al nivel de la microinformática individual;
- la exigencia de un acercamiento cada vez más "amigable" (user friendly) ha contribuido fuertemente a realizar interfaces de estas características amigables mucho más en ordenadores personales que en los grandes sistemas; la nueva generación de ordenadores personales, con tecnología avanzada, acentúa esta positiva tendencia (mientras el mismo Unix presenta a menudo una semántica de comandos mucho menos sencilla y clara que la que está en uso en los ordenadores personales).

Esperamos que con este libro tengan un poco más claro qué son y para qué sirven esos grandes desconocidos que son los sistemas operativos y todos los programas de soporte, contribuyendo así también a hacer más claro el entendimiento y diálogo con los ordenadores.

BIBLIOTECA BASICA INFORMATICA

INDICE GENERAL

10 **Sistemas operativos y software de base**
Qué son, para qué sirven. Unos desconocidos muy importantes.
11 **Sistema operativo CP/M**
Uno de los sistemas operativos para microprocesadores de 8 bits de mayor difusión en el mercado.
12 **MS-DOS: el estándar de IBM**
Sistema operativo para el microprocesador de 16 bits 8088, adoptado por el IBM-PC.
13 **Paquetes de aplicaciones. Software "pret a porter"**
Características y peculiaridades de los más importantes paquetes de aplicaciones.
14 **VisiCalc: una buena hoja de cálculo**
Interioridades y manejo de una de las hojas de cálculo más usadas.
15 **Dibujar con el ordenador**
Profundizando en una de las facetas útiles y divertidas que nos ofrecen los ordenadores.
16 **Tratamiento de textos... para escribir con el ordenador**
Cómo convertir su ordenador en una máquina de escribir con memoria y todo tipo de posibilidades.
17 **Diseño de juegos**
Particularidades características de esta aplicación de los ordenadores.
18 **LOGO: la tortuga inteligente**
Un lenguaje conocido por su «cursor gráfico», la tortuga, y sus aplicaciones pedagógicas al alcance de su mano.
19 **BASIC y tratamiento de imágenes**
Todo lo que en **¡Y llegó el BASIC!** no se pudo ver sobre las imágenes y gráficos en el BASIC.
20 **Bancos de datos (I)**
Peculiaridades de una de las aplicaciones de los ordenadores más interesantes, y que más dinero mueven.
21 **Bancos de datos (II)**
Profundizando en sus características.
22 **Paquetes integrados: Lotus 1-2-3 y Simphony**
Estudio de dos de los paquetes integrados (Hoja de cálculo + base de datos + ...) más conocidos.
23 **dBASE II y dBASE III**
Cómo aprovechar las dos versiones más recientes de esta importante base de datos.
24 **Los ordenadores uno a uno**
Un amplio y completo estudio comparativo.
25 **Cálculo numérico en BASIC**
Una aplicación especializada a su disposición.
26 **Multiplan**
Cómo hacer uso de este moderno paquete de aplicaciones.
27 **FORTRAN y COBOL**
Dos lenguajes muy especializados y distintos.
28 **FORTH: anatomía de un lenguaje inteligente**
Principales características de un lenguaje moderno, flexible y de amplio uso, en la robótica.
29 **Cómo realizar nuestro propio banco de datos**
Conocimientos necesarios para poder fabricar un banco de datos a nuestro gusto y medida.
30 **Los paquetes integrados uno a uno**
Todos los que usted puede encontrar en el mercado.

NOTA: Ingelek, S. A. se reserva el derecho de modificar, sin previo aviso, el orden, título o contenido de cualquier volumen de la colección.

NOTAS

Muchas veces podremos observar que, a la hora de hablar de un ordenador, no aparece por ningún lado la mención del sistema operativo. ¿Supone esto que no tiene ninguna importancia o bien es un fallo u olvido del vendedor?

Realmente no todo el mundo sabe qué es y para qué sirve un sistema operativo y, sin embargo, su influencia es tan grande que puede hacer que programas existentes en el mercado (de juegos o de aplicaciones) no puedan ejecutarse en su ordenador por disponer en él de un sistema operativo distinto al que necesita el programa para funcionar correctamente.

Lo mismo se podría decir de elementos como el editor de textos, compilador, cargador, enlazador..., programas de soporte que, junto al sistema operativo y los lenguajes de programación, constituyen el software de base o del sistema.

395 pts.
(incluido IVA)